

Jianyong Wang Hui Xiong
Yoshiharu Ishikawa Jianliang Xu
Junfeng Zhou (Eds.)

LNCS 7923

Web-Age Information Management

14th International Conference, WAIM 2013
Beidaihe, China, June 2013
Proceedings



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbruecken, Germany

Jianyong Wang Hui Xiong
Yoshiharu Ishikawa Jianliang Xu
Junfeng Zhou (Eds.)

Web-Age Information Management

14th International Conference, WAIM 2013
Beidaihe, China, June 14-16, 2013
Proceedings

Volume Editors

Jianyong Wang

Tsinghua University, Beijing, China

E-mail: jianyong@tsinghua.edu.cn

Hui Xiong

Rutgers, The State University of New Jersey

Newark, NJ, USA

E-mail: hxiong@rutgers.edu

Yoshiharu Ishikawa

Nagoya University, Japan

E-mail: ishikawa@is.nagoya-u.ac.jp

Jianliang Xu

Hong Kong Baptist University, China

E-mail: xujl@comp.hkbu.edu.hk

Junfeng Zhou

Yanshan University, Qinhuangdao, China

E-mail: zhoujf@ysu.edu.cn

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-38561-2

e-ISBN 978-3-642-38562-9

DOI 10.1007/978-3-642-38562-9

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013939515

CR Subject Classification (1998): H.2.4, H.2.7-8, H.3.3-5, F.2.2, H.2, H.4, C.2, H.5, G.2.2, I.5.3

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the proceedings of the 14th International Conference on Web-Age Information Management (WAIM), held during June 14–16, 2013, in Beidaihe, China. As a flagship conference in the Asia-Pacific region focusing on the research, development, and applications of Web information management, its success has been witnessed through the previous conferences in the series that were held in Shanghai (2000), Xi'an (2001), Beijing (2002), Chengdu (2003), Dalian (2004), Hangzhou (2005), Hong Kong (2006), Huangshan (2007), Zhangjiajie (2008), Suzhou (2009), Jiuzhaigou (2010), Wuhan (2011), and Harbin (2012). With the fast development of Web-related technologies (especially in emerging areas of social Web and big data in recent years), we expect that WAIM will become an increasingly popular forum to bring together outstanding researchers in this field from all over the world.

The WAIM 2013 Program Committee received 248 submissions from 16 countries and regions, from which we selected 46 full papers and 29 short papers for publication. The acceptance rate for regular papers is about 18.5%, while the overall acceptance rate is around 30.2% including short papers. The contributed papers address a wide range of topics, such as data mining, query processing and optimization, security, privacy, trust, recommender systems, information retrieval, spatial and temporal databases, information extraction, big data management and analysis, information integration and heterogeneous systems, Web search and meta-search, and data and information quality, among others.

As the 14th event in this increasingly popular series, WAIM 2013 featured a new program, i.e., the Distinguish Young Lecture (DYL) series. In this program, five speakers were invited—Hong Cheng, Xin Luna Dong, Bingsheng He, Guoliang Li, and Lei Yu—to give invited talks and chair sessions, and their invited research papers are also included in the proceedings. The panel of WAIM 2013 was on a salient topic of “Big Data Beyond MapReduce”, which emphasizes the challenges introduced by big data, from computational theory to effective techniques for querying big data. This year’s WAIM proudly presented five distinguished keynote speakers, Philip Yu, Wen Gao, Michael Franklin, Ming-Syan Chen, and Lei Chen, to share their visions and challenging research issues. In addition, the conference proceedings also contains a demo session with six accepted papers included.

Finally, we would like to express our gratitude to all the contributors who made this high-quality program possible. We gratefully acknowledge the support from National Natural Science Foundation of China and K.C. Wong Education Foundation, Hong Kong, and thank all the authors who chose WAIM as a venue for their publications. We are extremely indebted to the 133 Program Committee members and the external reviewers for their invaluable efforts. Without their timely and high-quality reviews, it would have not been possible to have such

a solid program. Last but not least, we would like to thank all the WAIM 2013 Organizing Committee members who gave tremendous help to our work, and sincerely hope that WAIM will continue its excellence and serve as an important forum for researchers to share their original research results in the field of Web information management.

Xiaofeng Meng

Huan Liu

Hiroyuki Kitagawa

Jianyong Wang

Hui Xiong

Yoshiharu Ishikawa

Conference Organization

General Co-chairs

Xiaofeng Meng	Renmin University of China, China
Huan Liu	Arizona State University, USA
Hiroyuki Kitagawa	University of Tsukuba, Japan

Program Committee Co-chairs

Jianyong Wang	Tsinghua University, China
Hui Xiong	Rutgers, The State University of New Jersey, USA
Yoshiharu Ishikawa	Nagoya University, Japan

Workshop Co-chairs

Yunjun Gao	Zhejiang University, China
Kyuseok Shim	Seoul National University, South Korea

Demonstration Co-chairs

Zhiguo Gong	Macau University, China
Peiquan Jin	University of Science and Technology of China, China
Takahiro Hara	Osaka University, Japan

Panel Chair

Wenfei Fan	Edinburgh University, UK
------------	--------------------------

DYL Series Co-chairs

Sean Xiaoyang Wang	Fudan University, China
Jeffrey Xu Yu	Chinese University of Hong Kong, China

Industry Co-chairs

Zhenkun Yang	Taobao Co., China
Mukesh K. Mohania	IBM Research India, India

Publication Co-chairs

Jianliang Xu Hong Kong Baptist University, China
Junfeng Zhou Yanshan University, China

Publicity Co-chairs

Xuan Zhou Renmin University of China, China
Wenjun Zhou University of Tennessee-Knoxville, USA
Hua Wang University of Southern Queensland, Australia

Local Arrangements Co-chairs

Jiadong Ren Yanshan University, China
Jingfeng Guo Yanshan University, China

Finance Co-chairs

Howard Leung City University of Hong Kong
Yukun Li Tianjin University of Technology, China

Registration Co-chairs

Ziyang Chen Yanshan University, China
Wei Cao Renmin University of China, China

Best Paper Award Co-chairs

Masaru Kitsuregawa University of Tokyo, Japan
Philip Yu University of Illinois at Chicago
Lizhu Zhou Tsinghua University, China

Steering Committee Liaison

Qing Li City University of Hong Kong, China

CCF DBS Liaison

Changjie Tang Sichuan University, China

APWeb Liaison

Xuemin Lin University of New South Wales, Australia

Microsoft Summer School Lian

Haixun Wang

Microsoft Research Asia, China

Program Committee

Toshiyuki Amagasa	University of Tsukuba, Japan
Ning An	Hefei University of Technology, China
Wolf-Tilo Balke	L3S Research Center, Germany
Yi Cai	City University of Hong Kong, China
Enhong Chen	University of Science and Technology of China
Gang Chen	Zhejiang University, China
Hong Chen	Renmin University of China
Lei Chen	Hong Kong University of Science and Technology, China
Weizhu Chen	Microsoft Research Asia
Ling Chen	University of Technology, Sydney, Australia
Yueguo Chen	Renmin University of China
Hong Cheng	Chinese University of Hong Kong, China
Reynold Cheng	The University of Hong Kong, China
David Cheung	The University of Hong Kong, China
Gao Cong	Nanyang Technological University, Singapore
Bin Cui	Peking University, China
Alfredo Cuzzocrea	University of Calabria, Italy
Ting Deng	Beihang University, China
Zhiming Ding	Institute of Software, Chinese Academy of Sciences, China
Guozhu Dong	Wright State University, USA
Dejing Dou	University of Oregon, USA
Aimin Feng	Nanjing University of Aeronautics and Astronautics, China
Yaokai Feng	Kyushu University, Japan
Johann Gamper	Free University of Bozen-Bolzano, Italy
Bryon Gao	Texas State University at San Marcos, USA
Hong Gao	Harbin Institute of Technology, China
Jun Gao	Peking University, China
Yong Gao	University of British Columbia, Canada
Yunjun Gao	Zhejiang University, China
Yong Ge Rutgers	University, USA
Xueqing Gong	East China Normal University, China
Zhiguo Gong	University of Macau, China
Giovanna Guerrini	Universita di Genova, Italy
Theo Haerder	University of Kaiserslautern, Germany
Binsheng He	Nanyang Technological University, Singapore
Xiaofeng He	East China Normal University, China

Luke Huan	University of Kansas, USA
Jianbin Huang	Xidian University, China
Jimmy Huang	York University, Canada
Xuanjing Huang	Fudan University, China
Leong Hou	University of Macau, Macau
Yan Jia	National University of Defense Technology, China
Lili Jiang	Max-Planck-Institut für Informatik, Germany
Peiquan Jin	University of Science and Technology of China
Wu Kui	Victoria University, Canada
Wookey Lee	Inha University, Korea
Carson K. Leung	University of Manitoba, Canada
Chengkai Li	University of Texas at Arlington, USA
Cuiping Li	Renmin University of China
Feifei Li	University of Utah, USA
Guohui Li	Huazhong University of Science and Technology, China
Guoliang Li	Tsinghua University, China
Qingzhong Li	Shandong University, China
Tao Li	Florida International University, USA
Xiaoli Li	Institute for Infocomm Research, Singapore
Xiang Lian	Hong Kong University of Science and Technology, China
Hongyan Liu	Tsinghua University, China
Jixue Liu	University of South Australia, Australia
Mengchi Liu	Carleton University, Canada
Qi Liu	University of Science and Technology of China, China
Qizhi Liu	Nanjing University, China
Hongan Lu	La Trobe University, Australia
Jiaheng Lu	Renmin University of China
Jizhou Luo	Harbin Institute of Technology, China
Ping Luo	HP Research Labs China
Xiangfeng Luo	Shanghai University, China
Shuai Ma	Beihang University, China
Sanjay Madria	Missouri University of Science and Technology, USA
Weiyi Meng	State University of New York at Binghamton, USA
Mukesh K. Mohania	IBM India Research Lab, India
Yang-Sae Moon	Kangwon National University, Korea
Maybin Muyeba	Manchester Metropolitan University, UK
Akiyo Nadamoto	Konan University, Japan
Shinsuke Nakajima	Kyoto Sangyo University, Japan
Anne Ngu	Texas State University at San Marcos, USA
Baoning Niu	Taiyuan University of Technology, China

Hiroaki Ohshima	Kyoto University, Japan
Makoto Onizuka	NTT, Japan
Jian Pei	Simon Fraser University, Canada
Wen-Chih Peng	National Chiao Tung University, Taiwan, China
Zhiyong Peng	Wuhan University, China
Tieyun Qian	Wuhan University, China
Weining Qian	East China Normal University, China
Xiaolin Qin	Nanhang University, China
Ning Ruan	Kent State University, USA
Heng Tao Shen	University of Queensland, Australia
Wei Shen	Tsinghua University, China
Kyuseok Shim	Seoul National University, Korea
Lidan Shou	Zhejiang University, China
Dandan Song	Beijing Institute of Technology, China
Heli Sun	Xi'an Jiaotong University, China
Weiwei Sun	Fudan University, China
Chih-Hua Tai	National Taipei University, Taiwan, China
Yufei Tao	Chinese University of Hong Kong, China
Taketoshi Ushijima	Kyushu University, Japan
Bin Wang	Northeast University, China
Chaokun Wang	Tsinghua University, China
Hua Wang	University of Southern Queensland, Australia
Hongzhi Wang	Harbin Institute of Technology, China
Jianmin Wang	Tsinghua University, China
Jing Wang	New York University, USA
Peng Wang	Fudan University, China
Wei Wang	Fudan University, China
Wei Wang	University of New South Wales, Australia
Yijie Wang	National University of Defense Technology, China
Zhongyuan Wang	Microsoft Research Asia
Qiang Wei	Tsinghua University, China
Junjie Wu	Beihang University, China
Sen Wu	University of Science and Technology Beijing, China
Shengli Wu	Jiangsu University, China
Xintao Wu	University of North Carolina at Charlotte, USA
Yuqing Wu	Indiana University at Bloomington, USA
Yanghua Xiao	Fudan University, China
Jierui Xie	Oracle Corporation, USA
Chunxiao Xing	Tsinghua University, China
Jianliang Xu	Hong Kong Baptist University, China
Xifeng Yan	University of California, Santa Barbara, USA
Lianghuai Yang	Zhejiang University of Technology, China
Ning Yang	Sichuan University, China
Xiaochun Yang	Northeast University, China

Yin Yang	Advanced Digital Sciences Center, Singapore
Ke Yi	Hong Kong University of Science and Technology, China
Jian Yin	Zhongshan University, China
Ge Yu	Northeast University, China
Hwanjo Yu	Pohang University of Science and Technology, Korea
Jeffrey Yu	Chinese University of Hong Kong, China
Philip Yu	University of Illinois at Chicago, USA
Ting Yu	North Carolina State University, USA
Xiaohui Yu	Shandong University, China
Hua Yuan	University of Electronic Science and Technology of China
Nicholas Jing	Yuan Microsoft Research Asia, China
Ming Zhang	Peking University, China
Nan Zhang	The George Washington University, USA
Xiangliang Zhang	King Abdullah University of Science and Technology (KAUST), Saudi Arabia
Zhiqiang Zhang	Harbin Engineering University, China
Xujian Zhao	Southwest University of Science and Technology, China
Ying Zhao	Tsinghua University, China
Baoyao Zhou	EMC Labs China
Junfeng Zhou	Yanshan University
Shuigeng Zhou	Fudan University, China
Wenjun Zhou	University of Tennessee-Knoxville, USA
Xuan Zhou	Renmin University of China
Feida Zhu	Singapore Management University
Xingquan Zhu	University of Technology, Sydney
Fuzhen Zhuang	ICT, Chinese Academy of Sciences, China
Yi Zhuang	Zhejiang Gongshang University, China
Lei Zou	Peking University, China
Zhaonian Zou	Harbin Institute of Technology, China

External Reviewers

Xue Bai	Fudan University, China
Lakshika Balasuriya	Wright State University, USA
Bokai Cao	Renmin University of China
Chong Chen	Fudan University, China
Chunan Chen	Fudan University, China
Lei Chen	Hong Kong Baptist University, China
Liu Chen	Wuhan University, China
Qinghao Dai	EMC Labs China
Lei Duan	Sichuan University, China

Shen Gao	Hong Kong Baptist University, China
Weihua Gong	Zhejiang University of Technology, China
Wei He	Renmin University of China
Juhua Hu	Simon Fraser University, Canada
Jeffrey Jesters	University of Utah, USA
Fan Jiang	University of Manitoba, Canada
Yu Jiang	Binghamton University, USA
Wangchao Le	University of Utah, USA
Siyu Lei	University of Hong Kong, China
Xian Li	Binghamton University, USA
Yafei Li	Hong Kong Baptist University, China
Yang Li	University of California at Santa Barbara, USA
Ye Li	Renmin University of China
Yijuan Lu	Texas State University-San Marcos, USA
Siqiang Luo	Fudan University, China
Yifeng Luo	Fudan University, China
Yongfang Ma	Tsinghua University, China
Silviu Maniu	University of Hong Kong, China
Xiangbo Mao	Simon Fraser University, Canada
Kenta Oku	Ritsumeikan University, Japan
Jian Pan	Zhejiang University of Technology, China
Yifan Pan	Indiana University, USA
Rodion Podorozhny	Texas State University-San Marcos, USA
Wei Song	Wuhan University, China
Guangfu Sun	University of Science and Technology of China
Huan Sun	University of California at Santa Barbara, USA
Liyang Tang	Hefei University of Technology, China
Ying Tang	University of California at Santa Barbara, USA
Tao Wang	South China University of Technology, China
Yue Wang	University of North Carolina at Charlotte, USA
Zhiang Wu	Nanjing University of Finance and Economics, China
Nana Xu	Renmin University of China
Chong Yang	Shandong University, China
Hedong Yang	Tsinghua University, China
Shawn Yang	University of Hong Kong, China
Ting Yu	Wuhan University, China
Ziqiang Yu	Shandong University, China
Jianwei Zhang	Tsukuba University of Technology, Japan
Lei Zhang	University of Science and Technology of China
Qun Zhao	Renmin University of China
Shuai Zhao	Tsinghua University, China
Haofeng Zhou	IBM
Bo Zong	University of California at Santa Barbara, USA

On Mining Big Data

Philip S. Yu

Department of Computer Science, The University of Illinois, USA

Abstract. The problem of big data has become increasingly importance in recent years. On the one hand, the big data is an asset that potentially can offer tremendous value or reward to the data owner. On the other hand, it poses tremendous challenges to realize the value out of the big data. The very nature of the big data poses challenges not only due to its volume, and velocity of being generated, but also its variety and veracity. Here variety means the data collected from various sources can have different formats from structured data to text to network/graph data to image, etc. Veracity concerns the trustworthiness of the data as the various data sources can have different reliability. In this talk, we will discuss these issues and approaches to address them.

AMPLab: Making Sense of Big Data with Algorithms, Machines and People

Michael Franklin

University of California, Berkeley

Abstract. The Berkeley AMPLab is developing a new data analysis software stack by deeply integrating machine learning and data analytics at scale (Algorithms), cloud and cluster computing (Machines) and crowdsourcing (People) to make sense of massive data. Current application efforts focus on cancer genomics, real-time traffic sensing and prediction, urban planning, and collaborative energy bug detection on smartphones. Launched in 2011, the AMPLab is a six-year effort sponsored in part by 21 leading technology companies including founding sponsors Amazon Web Services, Google and SAP. In March 2012 AMPLab was awarded a 5-year NSF CISE Expeditions in Computing award, which was featured as part of the 2012 White House Big Data research initiative. In addition to this broad support from both industry and government, the AMPLab is distinguished by its collaborative environment, where researchers across many different areas of computer science work together to develop new approaches to Big Data processing. In this talk I'll give an overview of the AMPLab structure and goals, and then focus on several subprojects, including the open source Spark and Shark processing frameworks, which have been shown to provide more than 100x performance improvement over the popular Hive system, and several projects on crowdsourced query processing. The focus will be on the new research opportunities that arise when looking beyond the traditional topic boundaries that have developed in and around the Computer Science discipline.

Towards Model-Based Video Coding and Analysis

Wen Gao

Peking University

Abstract. Image and video data is becoming the majority in big data era, a reasonable improvement of video coding efficiency may get a big cost saving in video transmission and/or storage, that is why so many researchers working on the new coding technologies and standards. For example, a joint video team between ISO and ITU-T works on HEVC standard which achieves about 50% bits saving over H.264/MPEG-4 AVC with the comparable visual quality, and a team under IEEE standard association works on a new standard IEEE 1857 for internet/surveillance video coding. However, the limitation of the further performance improvement under the traditional hybrid coding framework is recognized, researchers are looking for new model/technology. Recently, the model based coding (MBC) has achieved some exciting advancements on several image/video applications, such as model based screen video coding, surveillance video coding and cloud images coding etc, which is very promising for the next generation image/video coding. In this talk, the recent developments of model-based video coding will be given, special on background picture model based surveillance coding and cloud-based image coding, and the challenging problems and future directions of MBC will be discussed also.

On Application-Aware Graph Abstraction for Social Networks

Ming-Syan Chen

Research Center of Information Technology Innovation,
Academia Sinica, Taiwan

Abstract. Recently due to the fast increasing use of social networks, it has become very desirable to conduct various analyses/testing for applications on social networks. However, as the scale of a social network has become prohibitively large, it is infeasible to analyze the huge graph used to model the social network. As a result, a significant amount of research effort has been elaborated upon the abstraction of social network graphs. In essence, there are two approaches to graph abstraction, namely graph sampling and graph summarization. In this talk, we shall go over some recent studies on the graph sampling and graph summarization. We shall further explore the methods of summarization from the perspective of associated applications/queries. We shall show, in light of proper information processing, how guided queries can be efficiently carried out. A pattern-based graph generator will also be introduced. Finally, in addition to the common approaches for graph summarization which mainly rely upon the processing on graph structures, methods based on the progress of information spreading will be presented as an effective alternative for graph abstraction.

Crowdsourcing, a New Human Computing Paradigm?

Lei Chen

Hong Kong University of Science and Technology, Hong Kong
`leichen@cse.ust.hk`

Abstract. Recently, the popularity of crowdsourcing has brought a new opportunity to engage human intelligence into the process of various data analysis tasks. Compared with computer systems, crowds are good at handling items with human-intrinsic values or features. Existing work develop sophisticated methods by utilizing the crowd as a new kind of processors, a.k.a HPU (Human Processing Unit). As a consequence, tasks executed on HPU are called HPU-based tasks. At this point, a nature question arises: is crowdsourcing a new human computing paradigm?

In this talk, I will first discuss the answer to the above question and explain the key issues related to effective crowdsourcing. Then, I will demonstrate the power of crowdsourcing in solving the well-known and very hard data integration problem, schema matching. Finally, I will discuss how to migrate the power of corwdsourcing to a social media platform whose users can serve as a huge reservoir of workers and highlight some research challenges in this new human computing paradigm.

Table of Contents

Session 1: Data Mining (1)

Frequent Subgraph Summarization with Error Control	1
<i>Zheng Liu, Ruoming Jin, Hong Cheng, and Jeffrey Xu Yu</i>	
LSA-PTM: A Propagation-Based Topic Model Using Latent Semantic Analysis on Heterogeneous Information Networks	13
<i>Qian Wang, Zhaojun Peng, Fei Jiang, and Qingzhong Li</i>	
Improving Semi-supervised Text Classification by Using Wikipedia Knowledge	25
<i>Zhilin Zhang, Huaizhong Lin, Pengfei Li, Huazhong Wang, and Dongming Lu</i>	
Time Series Representation: A Random Shifting Perspective	37
<i>Xue Bai, Yun Xiong, Yangyong Zhu, and Hengshu Zhu</i>	
Mining Frequent Itemsets from Sparse Data Streams in Limited Memory Environments.....	51
<i>Juan J. Cameron, Alfredo Cuzzocrea, Fan Jiang, and Carson K. Leung</i>	

Human Dynamics Revealed through Log Analytics in a Cloud Computing Environment	58
<i>Sixi Chen, Ning An, Lian Li, Yongwei Wu, Weimin Zheng, and Lin Sun</i>	

Session 2: Information Integration and Heterogeneous Systems

Data Fusion: Resolving Conflicts from Multiple Sources	64
<i>Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava</i>	
Entity Resolution on Uncertain Relations	77
<i>Huabin Feng, Hongzhi Wang, Jianzhong Li, and Hong Gao</i>	
Imputation for Categorical Attributes with Probabilistic Reasoning.....	87
<i>Lian Jin, Hongzhi Wang, and Hong Gao</i>	
A New Approach to Identify Influential Spreaders in Complex Networks	99
<i>Qingcheng Hu, Yang Gao, Pengfei Ma, Yanshen Yin, Yong Zhang, and Chunxiao Xing</i>	

Presenting XML Schema Mapping with Conjunctive-Disjunctive Views	105
<i>Xuhui Li, Shafeng Zhu, Mengchi Liu, and Ming Zhong</i>	

An ETL Framework for Online Analytical Processing of Linked Open Data	111
<i>Hiroyuki Inoue, Toshiyuki Amagasa, and Hiroyuki Kitagawa</i>	

Session 3: Big Data

A Framework for Analyzing Monetary Cost of Database Systems in the Cloud.....	118
<i>Changbing Chen and Bingsheng He</i>	

Efficient Distributed Multi-dimensional Index for Big Data Management	130
<i>Xin Zhou, Xiao Zhang, Yanhao Wang, Rui Li, and Shan Wang</i>	

Fast Multi-fields Query Processing in Bigtable Based Cloud Systems ...	142
<i>Haiping Wang, Xiang Ci, and Xiaofeng Meng</i>	

ST-HBase: A Scalable Data Management System for Massive Geo-tagged Objects	155
<i>Youzhong Ma, Yu Zhang, and Xiaofeng Meng</i>	

EasyControl: Improve Database Throughput under Overloading	167
<i>Chengkai Tao, Qizhi Liu, and Yan Zhang</i>	

Combination of In-Memory Graph Computation with MapReduce: A Subgraph-Centric Method of PageRank	173
<i>QiuHong Li, Wei Wang, Peng Wang, Ke Dai, ZhiHui Wang, Yang Wang, and Weiwei Sun</i>	

Session 4: Spatial and Temporal Databases

A Human-Machine Method for Web Table Understanding	179
<i>Guoliang Li</i>	

Probabilistic k -Skyband Operator over Sliding Windows	190
<i>Xing Feng, Wenjie Zhang, Xiang Zhao, Ying Zhang, and Yunjun Gao</i>	

SATURN: A Fast Keyword k NN Search System in Road Networks	203
<i>Nan Zhang, Yang Wang, and Jianhua Feng</i>	

Top- K Aggregate Queries on Continuous Probabilistic Datasets	216
<i>Jianwen Chen, Ling Feng, and Jun Zhang</i>	

Shortest Path Finder with Light Materialized Path View for Location Based Services	229
<i>Aye Thida Hlaing, Htoo Htoo, Yutaka Ohsawa, Noboru Sonehara, and Masao Sakauchi</i>	

RUM+-tree: A New Multidimensional Index Supporting Frequent Updates	235
<i>Yuean Zhu, Shan Wang, Xuan Zhou, and Yansong Zhang</i>	

Session 5: Data Mining (2)

Joint Clustering and Feature Selection	241
<i>Liang Du and Yi-Dong Shen</i>	

A Self-Supervised Framework for Clustering Ensemble	253
<i>Liang Du, Yi-Dong Shen, Zhiyong Shen, Jianying Wang, and Zhiwu Xu</i>	

GCBN: A Hybrid Spatio-temporal Causal Model for Traffic Analysis and Prediction	265
<i>Chaogui Zhang and Jiangtao Ren</i>	

An Overlapped Community Partition Algorithm Based on Line Graph	277
<i>Zhenyu Zhang, Zhen Zhang, Wenzhong Yang, and Xiaohong Wu</i>	

Distance-Based Feature Selection from Probabilistic Data	282
<i>Tingting Zhao, Bin Pei, Suyun Zhao, Hong Chen, and Cuiping Li</i>	

Session 6: Data Mining (3)

Aspect-Specific Polarity-Aware Summarization of Online Reviews	289
<i>Gaoyan Ou, Wei Chen, Peng Liu, Tengjiao Wang, Dongqing Yang, Kai Lei, and Yueqin Liu</i>	

Fast Top- <i>k</i> Distance-Based Outlier Detection on Uncertain Data	301
<i>Salman Ahmed Shaikh and Hiroyuki Kitagawa</i>	

A Metric Learning Based Approach to Evaluate Task-Specific Time Series Similarity	314
<i>Yang Lu, Wayne Xin Zhao, Hongfei Yan, and Xiaoming Li</i>	

Using Coalitional Games to Detect Communities in Social Networks	326
<i>Lihua Zhou, Chao Cheng, Kevin Lü, and Hongmei Chen</i>	

A Comparison Study of Clustering Models for Online Review Sentiment Analysis	332
<i>Baojun Ma, Hua Yuan, and Qiang Wei</i>	

Session 7: Information Extraction

TopicDSDR: Combining Topic Decomposition and Data Reconstruction for Summarization	338
<i>Zhiming Zhang, Hongjie Li, and Lian'en Huang</i>	
CUVIM: Extracting Fresh Information from Social Network	351
<i>Rui Guo, Hongzhi Wang, Kaiyu Li, Jianzhong Li, and Hong Gao</i>	
Comments-Oriented Document Summarization Based on Multi-aspect Co-feedback Ranking	363
<i>Lifu Huang, Hongjie Li, and Lian'en Huang</i>	
Image Annotation with Weak Labels	375
<i>Feng Tian and Xukun Shen</i>	

Session 8: New Hardware and Miscellaneous

Scan and Join Optimization by Exploiting Internal Parallelism of Flash-Based Solid State Drives	381
<i>Wenyu Lai, Yulei Fan, and Xiaofeng Meng</i>	
MixSL: An Efficient Transaction Recovery Model in Flash-Based DBMS	393
<i>Yulei Fan and Xiaofeng Meng</i>	
Keyword Oriented Bitmap Join Index for In-Memory Analytical Processing	405
<i>Yansong Zhang, Mingchuan Su, Xuan Zhou, Shan Wang, and Xue Wang</i>	
Can SSDs Help Reduce Random I/Os in Hash Joins?	417
<i>Liang Huai Yang, Mingchao Liu, Yifan Pan, Weihua Gong, and Simon Stannus</i>	
An Index Model for Multitenant Data Storage in SaaS	423
<i>Pang Cheng, Li Qingzhong, and Kong Lanju</i>	

Session 9: Query Processing and Optimization

Reverse Top- k Group Nearest Neighbor Search	429
<i>Tao Jiang, Yunjun Gao, Bin Zhang, Qing Liu, and Lu Chen</i>	
Efficient Subsequence Search in Databases	440
<i>Rohit Jain, Mukesh K. Mohania, and Sunil Prabhakar</i>	
A Novel Data Broadcast Strategy for Traffic Information Query in the VANETs	453
<i>Xinjing Wang, Longjiang Guo, Jinbao Li, and Meirui Ren</i>	

A Spatial Proximity Based Compression Method for GML Documents	465
<i>Qingting Wei and Jihong Guan</i>	

Efficient MSubtree Results Computation for XML Keyword Queries	472
<i>Junfeng Zhou, Jinjia Zhao, Bo Wang, Xingmin Zhao, and Ziyang Chen</i>	

Session 10: Data Mining (4)

Measuring and Visualizing Interest Similarity between Microblog Users	478
<i>Jiayu Tang, Zhiyuan Liu, and Maosong Sun</i>	

Correlation Range Query	490
<i>Wenjun Zhou and Hao Zhang</i>	

Real Time Event Detection in Twitter	502
<i>Xun Wang, Feida Zhu, Jing Jiang, and Sujian Li</i>	

A Data Cleaning Framework Based on User Feedback	514
<i>Hui Xie, Hongzhi Wang, Jianzhong Li, and Hong Gao</i>	

Session 11: Social Network and Graphs

Finding Critical Blocks of Information Diffusion in Social Networks	521
<i>Ende Zhang, Guoren Wang, Kening Gao, and Ge Yu</i>	

Guide Query in Social Networks	533
<i>Yu-Chieh Lin, Philip S. Yu, and Ming-Syan Chen</i>	

Probabilistic Graph Summarization	545
<i>Nasrin Hassanlou, Maryam Shoaran, and Alex Thomo</i>	

Blind Chance: On Potential Trust Friends Query in Mobile Social Networks	557
<i>Jinzeng Zhang and Xiaofeng Meng</i>	

Sensitive Edges Protection in Social Networks	564
<i>Liangwen Yu, Tao Yang, Zhengang Wu, Jiawei Zhu, Jianbin Hu, and Zhong Chen</i>	

Session 12: Information Retrieval

Ontology-Based Semantic Search for Large-Scale RDF Data	570
<i>Xiaolong Tang, Xin Wang, Zhiyong Feng, and Longxiang Jiang</i>	

A Unified Generative Model for Characterizing Microblogs' Topics	583
<i>Kun Zhuang, Heyan Huang, Xin Xin, Xiaochi Wei, Xianxiang Yang, Chong Feng, and Ying Fang</i>	

A Novel Model for Medical Image Similarity Retrieval	595
<i>Pengyuan Li, Haiwei Pan, Jianzhong Li, Qilong Han, Xiaoqin Xie, and Zhiqiang Zhang</i>	

Finding Similar Questions with Categorization Information and Dependency Syntactic Tree	607
<i>Xin Lian, Xiaojie Yuan, Xiangyu Hu, and Haiwei Zhang</i>	

Ranking Web Pages by Associating Keywords with Locations	613
<i>Peiquan Jin, Xiaoxiang Zhang, Qingqing Zhang, Sheng Lin, and Lihua Yue</i>	

Session 13: Workflow Systems and Service Computing

Behavioral Consistency Measurement and Analysis of WS-BPEL Processes	619
<i>Xuewei Zhang, Wei Song, Jianchun Xing, Qiliang Yang, Hongda Wang, and Wenjia Zhang</i>	

Efficient Complex Event Processing under Boolean Model	631
<i>Shanglian Peng, Tianrui Li, Hongjun Wang, Jia He, Tiejun Wang, Fan Li, and An Liu</i>	

Bounding Trust under Uncertain Topology Information in Reputation-Based Trust Systems	643
<i>Xi Gong, Ting Yu, and Adam J. Lee</i>	

A Novel Service Selection Based on Resource-Directive Decomposition	649
<i>Dongming Jiang, Yuan Jiang, Wuping Xie, and Zhen You</i>	

Session 14: Recommender Systems

Collaborative Filtering Based on Rating Psychology	655
<i>Haijun Zhang, Chunyang Liu, Zhoujun Li, and Xiaoming Zhang</i>	

Modeling Semantic and Behavioral Relations for Query Suggestion	666
<i>Jimeng Chen, Yuan Wang, Jie Liu, and Yalou Huang</i>	

Mining User Interest and Its Evolution for Recommendation on the Micro-blogging System	679
<i>Jianjun Yu, Yi Shen, and Jianjun Xie</i>	

Collaborative Filtering Using Multidimensional Psychometrics Model ...	691
<i>Haijun Zhang, Xiaoming Zhang, Zhoujun Li, and Chunyang Liu</i>	

Incorporating Social Actions into Recommender Systems	698
<i>Di Ma, Dandan Song, and Lejian Liao</i>	

Session 15: Security, Privacy, and Trust

DiffR-Tree: A Differentially Private Spatial Index for OLAP Query	705
<i>Miao Wang, Xiaojian Zhang, and Xiaofeng Meng</i>	

K-core-preferred Attack to the Internet: Is it More Malicious Than Degree Attack?	717
<i>Jichang Zhao, Junjie Wu, Mingming Chen, Zhiwen Fang, and Ke Xu</i>	

Fuzzy Keyword Search over Encrypted Data in the Public Key Setting	729
<i>Qiuxiang Dong, Zhi Guan, Liang Wu, and Zhong Chen</i>	

The Hardness of (ϵ, m) -anonymity	741
<i>Yujia Li, Dong Li, Xianmang He, Wei Wang, and Huahui Chen</i>	

Session 16: Semantic Web and Ontology

Annotating Web Images by Combining Label Set Relevance with Correlation	747
<i>Feng Tian and Xukun Shen</i>	

Clustering Analysis and Semantics Annotation of 3D Models Based on Users' Implicit Feedbacks	757
<i>Lv Tianyang, Huang Shaobin, Wu Peng, and Lang Dapeng</i>	

bCATE: A Balanced Contention-Aware Transaction Execution Model for Highly Concurrent OLTP Systems	769
<i>Xiaogang Shi, Yanfei Lv, Yingxia Shao, and Bin Cui</i>	

Personalized News Recommendation Using Ontologies Harvested from the Web	781
<i>Junyang Rao, Aixia Jia, Yansong Feng, and Dongyan Zhao</i>	

TinyQP: A Query Processing System in Wireless Sensor Networks	788
<i>Shangfeng Mo, Hong Chen, Xiaoying Zhang, and Cuiping Li</i>	

CWePS: Chinese Web People Search	792
<i>Jianhua Yin and Lili Jiang</i>	

CONCERT: A Concept-Centric Web News Recommendation System	796
<i>Hongda Ren and Wei Feng</i>	

DOI Proxy Framework for Automated Entering and Validation of Scientific Papers	799
<i>Kun Ma, Bo Yang, and Guangwei Chen</i>	

Effectively Return Query Results for Keyword Search on XML Data	802
<i>Teng He, Guoqing Wu, Qingsong Chen, Junfeng Zhou, and Ziyang Chen</i>	
BrackitMR: Flexible XQuery Processing in MapReduce	806
<i>Caetano Sauer, Sebastian Bächle, and Theo Härdter</i>	
Author Index	809

Frequent Subgraph Summarization with Error Control

Zheng Liu¹, Ruoming Jin², Hong Cheng¹, and Jeffrey Xu Yu¹

¹ The Chinese University of Hong Kong

² Kent State University

{zliu,hcheng,yu}@se.cuhk.edu.hk,
jin@cs.kent.edu

Abstract. Frequent subgraph mining has been an important research problem in the literature. However, the huge number of discovered frequent subgraphs becomes the bottleneck for exploring and understanding the generated patterns. In this paper, we propose to summarize frequent subgraphs with an independence probabilistic model, with the goal to restore the frequent subgraphs and their frequencies accurately from a compact summarization model. To achieve a good summarization quality, our summarization framework allows users to specify an error tolerance σ , and our algorithms will discover k summarization templates in a top-down fashion and keep the frequency restoration error within σ . Experiments on real graph datasets show that our summarization framework can effectively control the frequency restoration error within 10% with a concise summarization model.

Keywords: frequent subgraph, pattern summarization.

1 Introduction

Frequent subgraph mining has been an important research problem in the literature, with many efficient algorithms proposed [10,12,21,24,1,8,17]. Given a collection \mathcal{D} of graphs, frequent subgraph mining is to discover all subgraphs whose frequencies are no less than a user-specified threshold f_{min} . Frequent subgraphs are useful in many applications, for example, as active chemical structures in HIV-screening datasets, spatial motifs in protein structural families, discriminative features in chemical compound classification [4], and indexing features [26] in graph databases to support graph queries.

One major issue in frequent subgraph mining is the difficulty of exploring and analyzing numerous patterns generated due to the exponential number of combinations. Given a graph with n edges, the total number of possible subgraphs could be 2^n . Hundreds of thousands of frequent subgraphs may be generated under a moderate minimum frequency threshold. A partial solution to this problem is mining closed or maximal frequent subgraphs [25,9,19], which generates fewer subgraph patterns. But in many cases the maximal and closed graph patterns can still be quite numerous, so the difficulty of exploring a large number of patterns still exists.

In the literature, there have been several methods which use probabilistic models to summarize frequent itemsets [23,22,11]. These probabilistic models, as a concise summarization, are effective to restore all the frequent itemsets and their frequencies. In this paper, we also use a probabilistic model to summarize frequent subgraphs by preserving their structure and frequency information as much as possible. Given a set of frequent subgraphs, we partition them into a few subsets. From each subset we choose a representative, which is called a template subgraph. All the frequent subgraphs in a subset are subgraphs of the corresponding template subgraph. The template subgraph is a summarization model which can restore the subgraphs in the subset and their frequencies. This problem is more challenging than itemset summarization, due to two difficulties in subgraph mining: (1) “multiple embeddings”, i.e., a subgraph can have multiple embeddings in a summarization model which is a template subgraph; and (2) “topological constraint”, i.e., the topological structure specifies the connectivity constraint among nodes and edges, while in itemset summarization, there is no such constraint between individual items. To solve the problem, we make an independence assumption between edges in a frequent subgraph. We take a regression approach to estimate the parameters in the independence probabilistic model by least square estimation. The probabilistic model allows users to restore frequent subgraphs and their frequencies from template subgraphs. To ensure a good summarization quality, we allow users to specify an error tolerance σ and our algorithms take a top-down approach to discover k template subgraphs. Multiple regression models will be built based on the k template subgraphs to control the frequency restoration error within σ . We have evaluated our subgraph summarization approach on real graph datasets. Experimental results show that our method can achieve a concise summarization with high accuracy in terms of subgraph frequency restoration error.

The rest of this paper is organized as follows. We formally define the problem of frequent subgraph summarization in Section 2 and propose the summarization algorithms in Section 3. We report the experimental results in Section 4 and discuss the related work in Section 5. Finally, Section 6 concludes the paper.

2 Problem Statement

A graph G is a triple (V, E, Γ) , where V and E are the node set and the edge set, respectively. Γ is a finite set of labels, and each node $v \in V$ or edge $(u, v) \in E$ is mapped to a label in Γ , denoted as $\Gamma(v)$ or $\Gamma(u, v)$. A graph g is a subgraph of another graph G if there exists a subgraph isomorphism from g to G , denoted as $g \subset G$. G is called a supergraph of g .

Definition 1. Subgraph Isomorphism. *For two graphs g and G , G contains a subgraph that is isomorphic to g , if there is an injective function $h : V_g \rightarrow V_G$, such that $\forall v \in V_g, \Gamma_g(v) = \Gamma_G(h(v))$, and $\forall (u, v) \in E_g, (h(u), h(v)) \in E_G$ and $\Gamma_g(u, v) = \Gamma_G(h(u), h(v))$, where Γ_g and Γ_G are the label sets of g and G , respectively.*

Definition 2. Frequent Subgraph. Given a collection \mathcal{D} of graphs, a graph g is frequent if $f(g) \geq f_{min}$, where $f(g)$ is the number of graphs in \mathcal{D} containing g , and f_{min} is a user-specified minimum frequency threshold.

A frequent subgraph g is a maximal one if and only if there does not exist another frequent subgraph g' and $g \subset g'$. A frequent subgraph g is a closed one if and only if there does not exist another frequent subgraph g' , $g \subset g'$ and $f(g') = f(g)$. Anti-monotonicity holds for frequent subgraphs in a graph database, which means the subgraphs of a frequent subgraph are also frequent.

Given a set of frequent subgraphs \mathcal{F} , we study how to summarize them into a small number of representatives, called template subgraphs, so that all frequent subgraphs and their frequencies can be restored accurately.

3 Summarization Algorithms

3.1 Subgraph Summarization by Regression

We propose to use a regression approach to frequent subgraph summarization. Based on the subgraph containment relationship, we can build a partial order graph (POG) consisting of all frequent subgraphs. Fig. 1 shows an example. Each node in the POG is a frequent subgraph. Subgraphs in the same level are of the same size, measured by the number of edges. In the POG, two subgraphs are connected by a directed edge from the larger one to the smaller one, if the smaller one is a subgraph of the larger and differs by one edge. Suppose g is a subgraph in the POG, we use connected children to represent its connected neighbors smaller than g , and g is called a connected parent. We use reachable children to represent all the nodes that can be reached by traveling along the edges in the POG, starting from g . A maximal subgraph does not have a connected parent in POG. If there is more than one maximal frequent subgraph, we add a union of all maximal frequent subgraphs as the root of the POG.

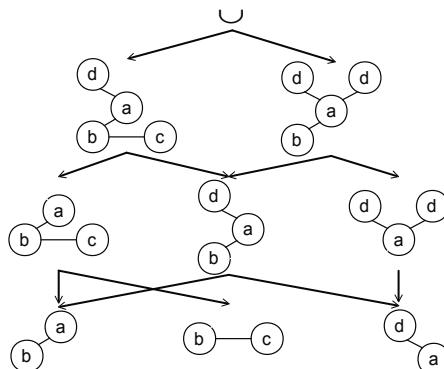


Fig. 1. Partial order graph of frequent subgraphs based on containment relationship

Suppose g and g' are two connected subgraphs in the POG, where g and g' differ by only one edge e , that is, $g \cup \{e\} = g'$. Let $p(g'|g)$ denote the conditional probability that a graph G from \mathcal{D} containing g also contains g' . Since $g \cup \{e\} = g'$, we denote $p(g'|g)$ as $p(e|g)$, the conditional probability that a graph G from a graph database \mathcal{D} containing g also contains edge e . Let $f(g)$ denote the frequency of a frequent subgraph g , then

$$p(g'|g) = p(e|g) = \frac{f(g')}{f(g)}. \quad (1)$$

Given any two frequent subgraphs g_1 and g_l that differ by $l - 1$ edges $\{e_1, e_2, \dots, e_{l-1}\}$, then

$$f(g_l) = f(g_1) \times p(e_1, e_2, \dots, e_{l-1}|g_1). \quad (2)$$

Let g_i denote the graph $g_1 \cup \{e_1, \dots, e_{i-1}\}$. Following the chain rule of conditional probabilities, we have

$$f(g_l) = f(g_1) \times \prod_{i=1}^{l-1} p(e_i|g_i). \quad (3)$$

To simplify the joint probability estimation, we apply the following independence assumption: whether a frequent subgraph g contains an edge e is independent of the structure of $g \setminus \{e\}$. Without loss of generality, we use $p(e)$ to denote $p(e|*)$, where $*$ denotes an arbitrary subgraph g that $g \cup \{e\}$ is frequent. Under this assumption, we can rewrite Eq. (3) for subgraph g_l and g_1 as follows:

$$f(g_l) = f(g_1) \times \prod_{i=1}^{l-1} p(e_i). \quad (4)$$

Given a frequent subgraph g , let $\mathcal{G} = \{g_1, g_2, \dots, g_n\}$ be the frequent subgraphs reachable from g in the POG. Suppose we know the frequency $f(g)$ of g , as well as all the probabilities $p(e_j)$ of edges in g , with the independent assumption, we can estimate the frequency of each graph $g_i \in \mathcal{G}$ according to Eq. (4). By applying the logarithmic transformation on both sides of the equation, we have

$$\log f(g) = \log f(g_i) + \sum_{j=1}^{i-1} \log p(e_j). \quad (5)$$

Similar to the regression approach in [11], we can build a regression model $Y = X\beta + E$ for \mathcal{G} , where E is the matrix of error terms,

$$Y = \begin{bmatrix} \log f(g) - \log f(g_1) \\ \dots \\ \log f(g) - \log f(g_n) \end{bmatrix}, X = \begin{bmatrix} \mathbf{1}_{e_1 \in g_1} & \dots & \mathbf{1}_{e_l \in g_1} \\ \dots & \dots & \dots \\ \mathbf{1}_{e_1 \in g_n} & \dots & \mathbf{1}_{e_l \in g_n} \end{bmatrix}, \text{ and } \beta = \begin{bmatrix} \log p(e_1) \\ \dots \\ \log p(e_l) \end{bmatrix}. \quad (6)$$

Here, $\mathbf{1}_{e_i \in g_j}$ is an indicator that edge e_i belongs to graph g_j . $\mathbf{1}_{e_i \in g_j} = 1$ if $e_i \in g_j$, and $\mathbf{1}_{e_i \in g_j} = 0$, otherwise. The least square estimation [18] of the above regression model is to minimize the sum of squares of the errors (residues), which is

$$\delta = \min_{\beta} \left\{ (Y - X\beta)'(Y - X\beta) \right\}. \quad (7)$$

Then the solution is

$$\begin{aligned} \hat{\beta} &= \arg \min_{\beta} \left\{ (Y - X\beta)'(Y - X\beta) \right\} \\ &= (X'X)^{-1}X'Y. \end{aligned} \quad (8)$$

By applying the above regression approach, we are able to summarize any frequent subgraph g in the POG, together with all its reachable children, as a regression model. We call g a template subgraph. The template subgraphs can restore all frequent subgraphs and their frequencies. We define the relative restoration error as follows.

Definition 3. Average Relative Restoration Error. Let \mathcal{F} denote the set of frequent subgraphs. For each subgraph $g \in \mathcal{F}$, $f(g)$ and $r(g)$ are the true frequency and the restored frequency of g , respectively. The relative frequency restoration error of g is $|r(g) - f(g)|/f(g)$, denoted as $\delta(g)$. The average relative restoration error of the frequent subgraph set \mathcal{F} is

$$\delta_{avg}(\mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_{g \in \mathcal{F}} \frac{|r(g) - f(g)|}{f(g)}. \quad (9)$$

Subgraph Summarization with Error Tolerance σ . Given a set of frequent subgraphs \mathcal{F} , and a maximum error tolerance σ , the problem of frequent subgraph summarization is to partition \mathcal{F} into as few groups as possible, and each group \mathcal{G} satisfies the following: (1) \mathcal{G} can be summarized as a single regression model, and (2) $\delta_{avg}(\mathcal{G}) \leq \sigma$, where δ_{avg} is defined in Definition 3.

3.2 Summarization Framework

Our summarization framework is presented in Algorithm 1, which is in a top-down fashion. The algorithm starts from a single template subgraph, the root of the POG, which is a union template subgraph for all maximal frequent subgraphs. Let g be a template subgraph, we use $\delta_{avg}(g)$ to denote the average restoration error of the regression model built on g with its reachable children. In each repeated loop from line 3 to line 8, the algorithm first divides the template subgraph with the maximum average restoration error into two template subgraphs, if the error is larger than the threshold σ , and then tries to merge the newly generated template subgraphs with other template subgraphs, until all the template subgraphs have an average restoration error $\leq \sigma$.

We use a binary tree T to maintain the generated template subgraphs. At line 1 in Algorithm 1, T is the root of POG \mathcal{G} . At line 6, we use the symbol of set

ALGORITHM 1: Summarization Framework

Input: POG \mathcal{G} , Error tolerance σ
Output: Template Subgraphs with Regression Models

```

1  $T \leftarrow$  the root of POG  $\mathcal{G}$ ;
2 while true do
3    $g' \leftarrow \arg \max_g \{\delta_{avg}(g) | g \in T\};$ 
4   if  $\delta_{avg}(g') > \sigma$  then
5      $\{g_1, g_2\} = \text{divide}(g');$ 
6      $T \leftarrow T \cup \{g_1, g_2\};$ 
7      $\text{merge}(T, g', g_1, g_2);$ 
8   end
9   else break;
10 end
11 return Template Subgraphs with Regression Models in T.
```

ALGORITHM 2: divide(Template g)

```

1  $C \leftarrow$  the directed children of  $g$  in POG  $\mathcal{G}$ ;
2  $\mathbf{Cand} \leftarrow \emptyset;$ 
3 foreach  $c_i \in C$  do
4   let  $G_i$  be the POG subgraph of  $g$  rooted at  $c_i$ ;
5   let  $G_g$  be  $g \setminus G_i$  /* $g$  corresponds to  $G_g$ */;
6   Build regression models  $R_i$  and  $R_g$  for  $G_i$  and  $G_g$  (or equivalently  $c_i$  and  $g$ );
7    $\epsilon_i \leftarrow$  total residue of  $R_i$ ;
8    $\epsilon_g \leftarrow$  total residue of  $R_g$ ;
9    $\mathbf{Cand} \leftarrow \mathbf{Cand} \cup \{(c_i, g, \epsilon_i + \epsilon_g)\};$ 
10 end
11  $(c_{min}, g_{min}) \leftarrow \arg \min_{c_i} \{\epsilon \mid (c_i, g, \epsilon) \in \mathbf{Cand}\};$ 
12 Update the regression models for  $c_i$  and  $g$  in  $\mathcal{G}$  based on  $c_{min}$  and  $g_{min}$ ;
13 return  $\{c_{min}, g_{min}\}$ 
```

union to denote the update of binary tree T . g_1 is a child of g' in POG, and g_2 is g' with a different regression model. The update is done by adding the new template subgraph g_1 to T as a child of g' and update the regression model of g' by the one of g_2 .

3.3 Template Subgraph Division

Algorithm 2 presents the procedure to divide a template subgraph. Consider a template graph g to be divided, the potential new template subgraphs are the connected children of g . Take Fig. 2 as an example. c_1 , c_2 and c_3 are g 's connected children in the POG. Suppose c_i is selected ($1 \leq i \leq 3$). Then we have two template subgraphs: c_i and g . There exist frequent subgraphs that are the descendants of both c_i and g . We restrict them to belong to only one

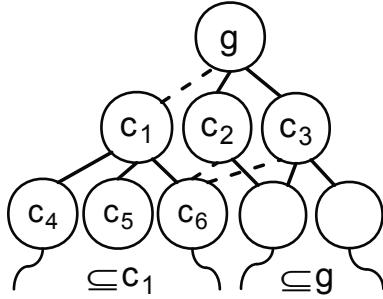


Fig. 2. Template Subgraph Division

template subgraph, either c_i or g , in order to obtain better regression models. The rule is to let the sharing part belong to the smaller template subgraph c_i .

We discuss how to build regression models for this case. Let e be the edge that appears in g but does not appear in c_i . All the descendants of c_i in the POG do not contain e . In other words, the descendants of g are divided into two parts: frequent subgraphs containing e and frequent subgraphs not containing e . By selecting c_i , the POG rooted at g is divided into two subgraphs. One POG subgraph G_i is rooted at c_i and contains all descendants of c_i . The other POG subgraph G_g contains g and all its descendants excluding those in G_i . As indicated in Fig. 2, the dotted lines indicate some descendant of G_g may be a supergraph of some graph in G_i because of the existence of the edge e , and must be deleted. We build regression models for G_i and G_g , respectively. Among all the possible children of g , we select the child node c_i of g in the POG which results in the minimum sum of residues of the regression models for both G_i and G_g . As the division continues, the residues and the average relative restoration errors will decrease. Eventually, the algorithm will stop when the average restoration error $\leq \sigma$.

3.4 Template Subgraph Merging

After the update of the binary tree for template subgraphs, a merging step is conducted at line 7 in Algorithm 1. The merging step serves as a refinement step of the binary tree with the hope to reduce the total number of template subgraphs by merging the updated template subgraphs with its parent subgraph or sibling subgraphs. Algorithm 3 presents the procedures of the merging step.

For a divided template subgraph g , g_1 is a child of g in the POG. Due to the division, all subgraphs reachable from g are divided into two sets, where g_1 and g_2 are the template subgraphs, respectively. Due to the change of the regression models, it is possible that g_2 could be merged with its parent g_p in the binary tree T and the average restoration error is below σ . Apparently, either the average restoration error of g_2 or that of g_p should be smaller than σ . For example, in Fig. 2, suppose c_1 is the divided template subgraph and c_4 is the child with the

ALGORITHM 3: merge(BinaryTree T , Template g, g_1, g_2)

```

1 if  $T$  contains only two template subgraphs then
2   | return;
3 end
4 Let  $g_p$  be the parent of template subgraph  $g$  in  $T$ ;
5 if  $lowerbound(g_p, g_2) \leq \sigma$  then
6   |  $\delta \leftarrow$  average restoration error of  $g_p$  after merging  $g_2$ ;
7   | if  $\delta \leq \sigma$  then
8     |   | Update  $g_p$  by merging  $g_p$  and  $g_2$ ;
9     |   | Remove  $g_2$  from  $T$ ;
10    | end
11 end

```

minimum total residue. Then it is possible that merging c_1 and g will generate a regression model whose average restoration error is below σ .

3.5 Querable Summarization

Given our frequent subgraph summarization with restoration error control, we can provide an answer when a user wants to know the frequency of a frequent subgraph. Since every frequent subgraph in the POG belongs to one and only one template subgraph, we only need to know which template subgraph it belongs to. Based on Eqs. (6) and (8), we can estimate the restored frequency $r(g)$ of a frequent subgraph g according to the following equation,

$$r(g) = x\hat{\beta}, \quad (10)$$

where $x = [\mathbf{1}_{e_1 \in g} \cdots \mathbf{1}_{e_l \in g}]$. Recall that $\mathbf{1}_{e_i \in g}$ is an indicator that edge e_i in the template graph belongs to graph g . $\mathbf{1}_{e_i \in g} = 1$ if $e_i \in g$, and $\mathbf{1}_{e_i \notin g} = 0$ if $e_i \notin g$. In our summarization framework, there is a partial order among edges to avoid the problems caused by multiple embeddings in a template graph. Upon all the template subgraphs obtained, we can identify which template subgraph a query graph g belongs to by utilizing the global edge ID's.

4 Experimental Results

In this section, we evaluate the performance of our proposed summarization method. All the experiments were run on a server with 4 CPU and 24GB memory running GNU/Linux.

We use the AIDS antiviral screen compound dataset¹. There are totally 43850 chemical compounds, which are classified into three categories: Confirmed Active (CA); Confirmed Moderately active (CM); and Confirmed Inactive (CI). As chemical compounds belonging to CI are not useful for the drug discovery, we

¹ http://dtp.nci.nih.gov/docs/aids/aids_data.html

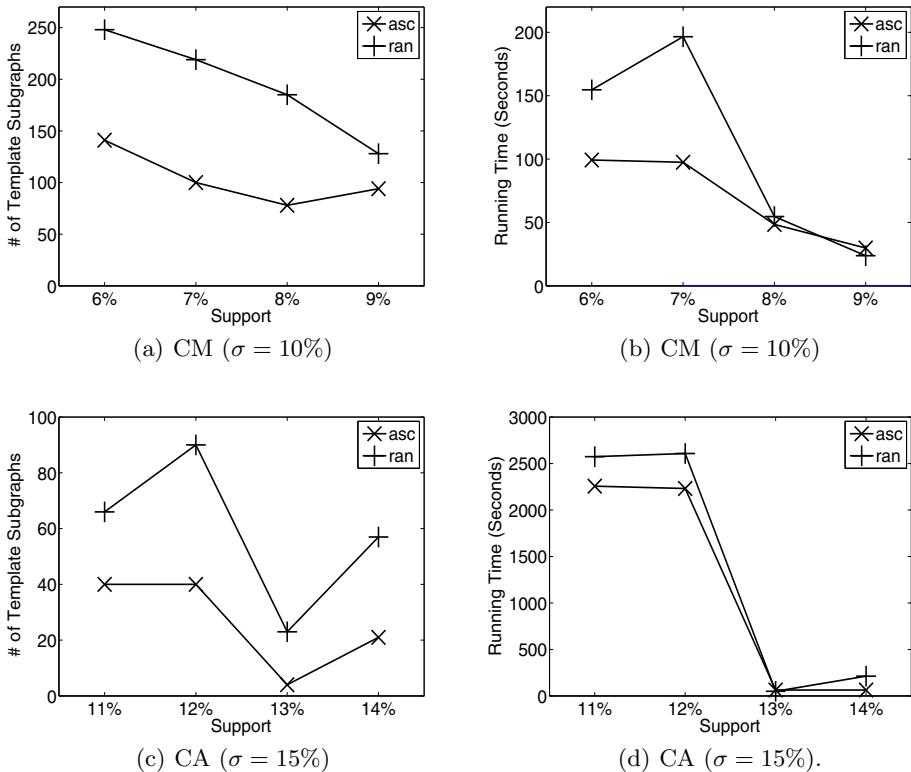


Fig. 3. Summarization Results on CM and CA

use datasets CA and CM in our experiments. CA contains 463 compounds and CM contains 1093 compounds. We list the number of frequent subgraphs mined from CA and CM under different minimum support in Table 1.

Experiment Setting: We implemented two variants of our summarization framework in Algorithm 1, namely, *asc* and *ran*. As discussed, in a union template subgraph, we could arrange the order of maximal frequent subgraphs in different ways to avoid the issues caused by multiple embeddings and common reachable children. *ran* denotes that we arrange the order randomly in division, while *asc* denotes that maximal frequent subgraphs are always sorted in the

Table 1. Number of Frequent Subgraphs

	CM				CA			
Minimum support	6%	7%	8%	9%	11%	12%	13%	14%
# of frequent subgraphs	5997	4265	3415	2627	15231	14318	8094	7612

ascending order of their sizes, measured by number of edges, during the union template creation.

Fig. 3 reports the summarization results on datasets CM and CA. Fig. 3(a) and 3(b) report the quality and running time on CM under different values of minimum support when the restoration error tolerance is 10%. Generally, a smaller value of support means a large number of frequent subgraphs. Under the same error tolerance, *asc* can generate a more compact summarization, i.e., a smaller number of template subgraphs, than *ran*. It is also more efficient than *ran*. Similar performance can be observed in Fig. 3(c) and 3(d) on CA. *acs* always generates fewer template subgraphs than *ran*. Compared with the number of frequent subgraphs in CM and CA shown in Table 1, we can find that the number of generated template subgraphs is up to several hundred times smaller than that of the frequent subgraphs in both datasets under a smaller error tolerance.

5 Related Works

Frequent Subgraph Summarization: One potential issue in frequent subgraph mining is the huge number of the discovered frequent subgraphs. Closed frequent subgraph [25] and maximal frequent subgraph [9,19] can partially solve this issue, but in many cases the number of closed or maximal subgraphs is still very huge. Recently researchers [3,15,28,6] have focused on selecting a small number of representative graph patterns to represent many similar subgraphs. The similarity could be maximum common subgraph [6], graph edit distance [3,28], or Jaccard distance [15]. Sampling is another approach to solve the overloading issue. Hasan et al. [7] proposed a sampling strategy based random walks on the frequent subgraph lattice.

Frequent Subgraph Mining: Many algorithms have been proposed for finding frequent subgraphs in graph databases, where the frequency of a subgraph is the total number of graphs containing the subgraph in the database. Similar to the Apriori-based approaches in frequent itemset mining, Apriori-based algorithms for frequent subgraph mining are proposed in [10,12,21], where the search strategy follows a breadth-first manner. Subgraphs of small sizes are searched first. Once identified, new candidate subgraphs are generated by joining two highly overlapping frequent subgraphs. In each iteration, the size of these candidate subgraphs is increased by one. Other algorithms [24,1,8,17] employ a pattern-growth style. New candidate subgraphs are generated by adding a new edge to the current. It is possible that a candidate subgraph can be extended from multiple frequent subgraphs. In gSpan [24], each candidate subgraph is associated with a depth-first code for duplicate identification. There are also research efforts on finding frequent subgraphs in a single large graph [2,14,5,13], where an important problem is how to calculate the frequency. One solution [13] is to consider the maximum number of non-overlapping embeddings as the frequency.

Other Large Graph Summarization: There are also works for large graph summarization. Navlakha et al. [16] proposed an approach to generate a compact

graph representation which can be restored to the original graph with bounded error. They used a single super-node to represent a clique or near-clique, with an additional table recording the missing edges. The quality of the summary is measured by the minimum description length. Tian et al. [20] proposed two aggregating algorithms for graph summarization. The top-down approach first groups together all the nodes with the same category attributes. Then the groups of nodes are repeatedly split until there are k groups. The bottom-up approach first groups together all the nodes with both the same node attributes and the same edge attributes. Then small groups of nodes are merged into larger ones till there are k groups left. Zhang et al. [27] extended Tian’s approach to deal with numerical attributes by automatically categorizing numerical attribute values. They also proposed an interestingness measure to identify the most interesting resolutions.

6 Conclusions

In this paper, we have proposed a frequent subgraph summarization framework with an independence probabilistic model. A regression approach is applied to estimate the parameters in the summarization model. Our summarization framework takes a top-down approach to recursively partition a summarization graph template into two, until the user-specified error tolerance is satisfied. Experimental results on real datasets show that our summarization model can effectively control the frequency restoration error within 10% with a concise representation.

In the future, we plan to study how to integrate our summarization framework into the pattern mining process to save the computation cost of finding all frequent subgraphs. We will also study the frequent subgraph summarization problem in a single large graph. Depending on the definition of frequency, the anti-monotonicity property does not always hold, which will introduce new challenges in the summarization framework to avoid false-positive subgraphs.

Acknowledgments. This work is supported by the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) Project No. CUHK 418512 and 411211.

References

1. Borgelt, C., Berthold, M.R.: Mining molecular fragments: Finding relevant substructures of molecules. In: ICDM, p. 51 (2002)
2. Bringmann, B., Nijssen, S.: What is frequent in a single graph? In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 858–863. Springer, Heidelberg (2008)
3. Chen, C., Lin, C.X., Yan, X., Han, J.: On effective presentation of graph patterns: a structural representative approach. In: CIKM, pp. 299–308 (2008)
4. Deshpande, M., Kuramochi, M., Wale, N., Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans. on Knowl. and Data Eng. 17(8), 1036–1050 (2005)

5. Fiedler, M., Borgelt, C.: Subgraph support in a single large graph. In: ICDM Workshops, pp. 399–404 (2007)
6. Hasan, M.A., Chaoji, V., Salem, S., Besson, J., Zaki, M.J.: Origami: Mining representative orthogonal graph patterns. In: ICDM, pp. 153–162 (2007)
7. Hasan, M.A., Zaki, M.J.: Output space sampling for graph patterns. Proc. VLDB Endow. 2(1), 730–741 (2009)
8. Huan, J., Wang, W., Prins, J.: Efficient mining of frequent subgraphs in the presence of isomorphism. In: ICDM, p. 549 (2003)
9. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: KDD, pp. 581–586 (2004)
10. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Źytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
11. Jin, R., Abu-Ata, M., Xiang, Y., Ruan, N.: Effective and efficient itemset pattern summarization: Regression-based approaches. In: KDD, pp. 399–407 (2008)
12. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM, pp. 313–320 (2001)
13. Kuramochi, M., Karypis, G.: Finding frequent patterns in a large sparse graph. Data Min. Knowl. Discov. 11(3), 243–271 (2005)
14. Li, S., Zhang, S., Yang, J.: DESSIN: Mining dense subgraph patterns in a single graph. In: Gertz, M., Ludäscher, B. (eds.) SSDBM 2010. LNCS, vol. 6187, pp. 178–195. Springer, Heidelberg (2010)
15. Liu, Y., Li, J., Gao, H.: Summarizing graph patterns. In: ICDE, pp. 903–912 (2008)
16. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph summarization with bounded error. In: SIGMOD, pp. 419–432 (2008)
17. Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: KDD, pp. 647–652 (2004)
18. Freund, R.J., Wilson, W.J., Sa, P.: Regression Analysis: Statistical Modeling of a Response Variable, 2nd edn. Academic Press (2006)
19. Thomas, L.T., Valluri, S.R., Karlapalem, K.: Margin: Maximal frequent subgraph mining. In: ICDM, pp. 1097–1101 (2006)
20. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: SIGMOD, pp. 567–580 (2008)
21. Vanetik, N., Gudes, E., Shimony, S.E.: Computing frequent graph patterns from semistructured data. In: ICDM, p. 458 (2002)
22. Wang, C., Parthasarathy, S.: Summarizing itemset patterns using probabilistic models. In: KDD, pp. 730–735 (2006)
23. Yan, X., Cheng, H., Han, J., Xin, D.: Summarizing itemset patterns: a profile-based approach. In: KDD, pp. 314–323 (2005)
24. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: ICDM, p. 721 (2002)
25. Yan, X., Han, J.: Closegraph: mining closed frequent graph patterns. In: KDD, pp. 286–295 (2003)
26. Yan, X., Yu, P.S., Han, J.: Graph indexing based on discriminative frequent structure analysis. ACM Trans. Database Syst. 30(4), 960–993 (2005)
27. Zhang, N., Tian, Y., Patel, J.: Discovery-driven graph summarization. In: ICDE, pp. 880–891 (2010)
28. Zhang, S., Yang, J., Li, S.: Ring: An integrated method for frequent representative subgraph mining. In: ICDM, pp. 1082–1087 (2009)

LSA-PTM: A Propagation-Based Topic Model Using Latent Semantic Analysis on Heterogeneous Information Networks

Qian Wang^{1,2}, Zhaohui Peng^{1,2,*}, Fei Jiang^{1,2}, and Qingzhong Li^{1,2}

¹ School of Computer Science and Technology, Shandong University, Jinan, China

² Shandong Provincial Key Laboratory of Software Engineering

{wangqian8636,jf29762}@gmail.com,
{pzh,lqz}@sdu.edu.cn

Abstract. Topic modeling on information networks is important for data analysis. Although there are many advanced techniques for this task, few methods either consider it into heterogeneous information networks or the readability of discovered topics. In this paper, we study the problem of topic modeling on heterogeneous information networks by putting forward LSA-PTM. LSA-PTM first extracts meaningful frequent phrases from documents captured from heterogeneous information network. Subsequently, latent semantic analysis is conducted on these phrases, which can obtain the inherent topics of the documents. Then we introduce a topic propagation method that propagates the topics obtained by LSA on the heterogeneous information network via the links between different objects, which can optimize the topics and identify clusters of multi-typed objects simultaneously. To make the topics more understandable, a topic description is calculated for each discovered topic. We apply LSA-PTM on real data, and experimental results prove its effectiveness.

Keywords: topic modeling, latent semantic analysis, topic propagation, heterogeneous information network.

1 Introduction

Information networks have been popularly used to represent the networked systems, such as social network, bibliographic network and so on. Data mining and knowledge discovery can be conducted on these networks [1]. Recently, with the explosion of textual documents in the heterogeneous information networks, such as papers, product reviews and other textual content, text-rich heterogeneous information networks come into being. Taking bibliographic data for example, one author can write several papers and one paper may be written by several authors. Likewise, each paper is commonly published in a venue (e.g., conferences, journals, etc.) and a venue usually publishes a number of papers. Thus we not only obtain the textual content of documents, but also the interactions among multi-typed objects as illustrated in Figure 1.

* Corresponding author.

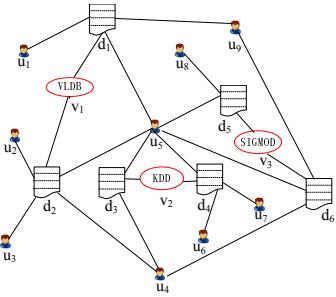


Fig. 1. Example of heterogeneous information network

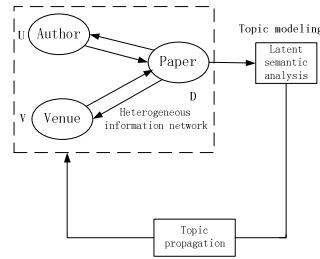


Fig. 2. Topic modeling on heterogeneous Information network

Topic modeling on information networks has been proved to be useful for document analysis. However, many topic models, such as LaplacianPLSI [2], NetPLSA [3] and iTopicmodel [4], merely deal with homogeneous networks. Besides, most existing methods [3, 4, 5, 6, 7] represent the discovered topics by word distributions. Although the topic word distributions may be intuitively meaningful, it is still difficult for users to fully understand the meaning of each topic. To address these problems, in this paper, we investigate and develop a novel topic model, LSA-PTM, for heterogeneous information networks, which solves the problems of topic modeling on heterogeneous information networks and the readability of discovered topics. LSA-PTM first extracts meaningful frequent phrases from documents that are captured from text-rich heterogeneous information network. The meaningful frequent phrases are useful to the readability of the discovered topics. Subsequently latent semantic analysis is conducted on these phrases, which can model the inherent topics of documents. Furthermore, we introduce a topic propagation method based on the links between different objects, which enhances the inherent topics and identify clusters of multi-typed objects simultaneously. The basic idea of the topic propagation is to propagate the topics obtained by topics models on the heterogeneous information networks as illustrated in Figure 2. To better understand the meaning of each topic, a topic description is computed for each topic.

In short, the contributions of our paper can be summarized as follows: (1) the proposed LSA-PTM effectively incorporates heterogeneous information networks with topic modeling; (2) LSA-PTM solves the intelligibility of each discovered topic by modeling each topic as a set of meaningful frequent phrases along with a topic description; (3) experiments are conducted on real dataset, and the results prove the effectiveness of our proposed model.

The rest of this paper is organized as follows. In section 2, we introduce some basic concepts. We elaborate LSA-PTM in Section 3. Section 4 presents the extensive experiment results. Finally, we discuss the related work in section 5 and conclude our work in section 6.

2 Basic Concepts

In this section, we formally introduce several related concepts and notations.

Definition 1. (Heterogeneous Information Network): A heterogeneous information network is defined as an information network with multiple types of objects and/or multiple types of links.

Text-rich Heterogeneous Information Network: When a heterogeneous information network contains a set of textual documents, it turns into a text-rich heterogeneous information network.

DBLP Bibliographic Network Example: As shown in Figure 1, there are three types of objects (i.e., $N = 3$) in this bibliographic network, including authors U , venues V and papers D . The bibliographic network can be denoted as $G = (D \cup U \cup V, E)$, where E is the set of edges that describe the relationships between papers $D = \{d_1, \dots, d_n\}$ and authors $U = \{u_1, \dots, u_l\}$ as well as venues $V = \{v_1, \dots, v_o\}$.

Definition 2. (Topic Description): A topic description is a phrase containing several sequential words which can largely cover the semantic meaning of the discovered topic. In our model, each discovered topic is represented by a set of meaningful frequent phrases, and the topic description is calculated from these phrases.

For example, if a discovered topic is represented by a set of meaningful frequent phrases, such as “database systems”, “database management”, “distributed databases”, “relational databases”, and so on. Through a serial of calculation, “database systems” could be the topic description of this topic.

3 LSA-PTM

In this section, we elaborate our proposed model, LSA-PTM, which is a propagation-based topic model using latent semantic analysis on heterogeneous information networks.

3.1 Meaningful Frequent Phrases Extraction

Phrases that consist of two to five sequential words that appear in more than ψ documents in the corpus are defined as frequent phrases. The threshold ψ is to filter the “noise” phrases that just appear in a handful of documents, and it is decided by the number of documents in document corpus. For example, frequent phrases can be “data mining”, “query processing”, and “data analysis” in data mining area.

In a preprocessing step, we use a sliding window over the document content to generate frequent phrases and lossy-counting [8] to discover the phrases with frequency above ψ , forming a frequent phrase list for each document. Each frequent phrase list contains the frequency that the phrase appears in this document.

Meaningful frequent phrases are defined as the frequent phrases that can describe the document corpus concisely. Frequent phrases extracted in preprocessing step contain a large list of stop word combinations and other meaningless phrases that fail

to capture the specific information in the documents. In order to get the meaningful frequent phrases, we exploit Algorithm 1 to weight the meaningful frequent phrases from the meaningless phrases appropriately.

Algorithm 1 summarizes the process of discovering meaningful frequent phrases for each document. In the processing step, a frequent phrase list has been formed for each document that contains the frequency of the phrases. We integrate these lists and add the frequency of the same phrases together to form an aggregate frequent phrase list, AFP-list. The input of Algorithm 1 is the frequent phrases in AFP-list, the number N_{con} of documents containing the phrase, total number N_{docs} of documents in document corpus, and the number m of meaningful frequent phrases that we want to compute. Our algorithm processes all the frequent phrases in AFP-list, and computes a relevance score with the formula in line 4. The formula boosts the meaningful frequent phrases, and filters the meaningless frequent phrases, with the similar idea with TF·IDF. Through this algorithm, we get the top m meaningful frequent phrases, (mfp_1, \dots, mfp_m).

Algorithm 1. Meaningful Frequent Phrases Extraction

Input: N_{con} : number of documents containing the phrase; N_{docs} : total number of documents in the corpus; m : number of meaningful frequent phrases; AFP-list

Output : queue: priority queue of <phrase, relevance>

- 1 **for** each frequent phrase in AFP-list **do**
- 2 phrase_f \leftarrow frequency of the frequent phrase in AFP-list
- 3 phrase_{len} \leftarrow sum of the phrase frequencies in AFP-list
- 4 relevance \leftarrow $\frac{\text{phrase}_f}{\text{phrase}_{len}} * \log \frac{N_{docs}}{N_{con}}$
- 5 insert <phrase, relevance> in queue
- 6 **if** queue has more than m items **then**
- 7 delete item from queue with the smallest relevance
- 8 **end if**
- 9 **end for**
- 10 **Return** the queue with m meaningful frequent phrases

3.2 LSA Based on Document-Phrase Matrix

LSA model [9] projects documents as well as terms into a low-dimensional semantic space by carrying out Singular Value Decomposition (SVD) [10], producing a set of topics associated with the documents and terms.

In our model, documents are represented as a set of meaningful frequent phrases extracted by Algorithm 1. Consider the analysis of document-phrase matrix, if there are a total of n documents and m meaningful frequent phrases in a document collection, phrase-document matrix, $A = [a_{ij}]_{m \times n}$, is constructed, with each entry a_{ij} representing the TF-IDF score of the i -th meaningful frequent phrase in the j -th document. For the matrix A , where without loss of generality $m \geq n$ and $\text{rank}(A) = r$, the SVD is defined as:

$$A = U \sum V^T \quad (1)$$

where $U = [u_1, u_2, \dots, u_r]$ is a $m \times r$ column-orthonormal matrix whose columns are called left singular vectors; $\Sigma = \text{diag}[\sigma_1, \sigma_2, \dots, \sigma_r]$ is an $r \times r$ diagonal matrix, whose diagonal elements are non-negative singular values sorted in descending order; $V = [v_1, v_2, \dots, v_r]$ is an $n \times r$ column-orthonormal matrix whose columns are called right singular vectors.

Given an integer k ($k \ll r$), LSA uses the first k singular vectors to represent the documents and meaningful frequent phrases in a same k -dimensional semantic space in which similar objects are expected to be near to each other. More precisely, each document is represented by a row of $[\sigma_1 v_1, \dots, \sigma_k v_k]$ and each meaningful frequent phrase is represented by a row of $[\sigma_1 u_1, \dots, \sigma_k u_k]$.

From the k -dimensional latent semantic space, we can get the inherent topics associated with papers and meaningful frequent phrases. LSA provides a simplified solution to model topics of documents in a text-rich heterogeneous information network. However, this model ignores the associated heterogeneous information network as well as other interacted objects, so it cannot model and make use of associated objects simultaneously.

3.3 Topic Propagation

In this section, we introduce a novel and general topic propagation method that effectively incorporates the heterogeneous information network with the textual documents for topic modeling, so as to estimate the topics of documents as well as the associated objects and improve the topic modeling results simultaneously.

To obtain the topics of other objects, a simple way is to propagate the topics from documents to other objects through the heterogeneous information network as shown in the dashed rectangle of Figure 2.

Principle. In a heterogeneous information network, the topic of an object without text content (e.g. u_1, v_1 in Figure 1) is decided by the topics of its connected documents. On the other hand, the topic of a document is affected by the estimated topics of its connected objects.

For example, the research topic of an author could be decided by his/her published papers. Conversely, the topic of a paper is influenced by its authors to a certain degree.

Definition 3. In k -dimensional semantic space, let matrix $[p(d_1), \dots, p(d_n)]^T$ represents the papers, where the values of k -dimensional vector $p(d_i), p(d_i)_j (j = 1 \dots k)$, represent the weights of paper d_i in the k topics; let matrix $[p(u_1), \dots, p(u_l)]^T$ represents the authors, where the values of k -dimensional vector $p(u_i), p(u_i)_j (j = 1 \dots k)$, represent the weights of author u_i in the k topics; let matrix $[p(v_1), \dots, p(v_o)]^T$ represents the venues, where the values of k -dimensional vector $p(v_i), p(v_i)_j (j = 1 \dots k)$, represent the weights of venue v_i in the k topics; let matrix $[p(mfp_1), \dots, p(mfp_m)]^T$ represents the meaningful frequent phrases, where the values of k -dimensional vector $p(mfp_i), p(mfp_i)_j (j = 1 \dots k)$, represent the weights of phrase mfp_i in the k topics.

Take Figure 1 for example, let us see how the topics propagate from papers to their neighboring objects, e.g., authors and venues. Given the initial vector of papers $p(d_i)$ in a certain k -dimensional topic space, the vector of an author $p(u_i)$ can be calculated by:

$$p(u_i) = \sum_{d_i \in D_u} \frac{p(d_i)}{|D_u|} \quad (2)$$

where D_u is the set of documents that are connected with author u , and $|D_u|$ is the number of these documents. Similarly, the vector of a venue $p(v_i)$ can be expressed as:

$$p(v_i) = \sum_{d_i \in D_v} \frac{p(d_i)}{|D_v|} \quad (3)$$

where D_v is the set of documents that are published in venue v .

On the other hand, let us see how the topics propagate from these objects without text content, e.g., authors and venues, to the papers, so as to reinforce the topics of papers. Along with the vectors of authors and venues calculated above, the vectors of papers can be reinforced by:

$$p(d_i) = \zeta p(d_i) + \frac{1 - \zeta}{2} \left(\sum_{u \in U_d} \frac{p(u_i)}{|U_d|} + \sum_{v \in V_d} \frac{p(v_i)}{|V_d|} \right) \quad (4)$$

where U_d is the set of authors of document d , and V_d is the venues connected with document d . Here, ζ is the harmonic parameter that controls the balance between the inherent topics and the propagated topics.

According to Eqs. (2), (3) and (4), we calculate the ultimate vectors of different objects, e.g., papers, authors and venues, in the k-dimensional latent semantic space through M iterations (M is a maximum iteration based on empirically study). The topic propagation process can be summarized as Algorithm 2. Note that, if $\zeta=1$, the topics of documents remain the original ones, while the topics of other objects are determined by their associated documents in one step.

Algorithm 2. Topic propagation

Input: $p(d_i)$: the vector of paper; $p(u_i)$: the vector of author; $p(v_i)$: the vector of venue; A : the set of authors; V : the set of venues; D : the set of papers; t : the number of iterations; M : a maximum iteration; D_u : the set of documents connecting with author u ; D_v : the set of documents published in venue v ; U_d : the set of authors of document d ; V_d : the venues connected with document d

Output: the ultimate vectors $p(d_i), p(u_i), p(v_i)$

1 The initial values of $p(d_i)^{(0)}$ is obtained by LSA; $t \leftarrow 1$

2 **do**

3 **for** each author in A and each venue in V **do**

$$4 \quad p(u_i)^{(t)} = \frac{1}{|D_u|} \sum_{d_i \in D_u} p(d_i)^{(t-1)}, \quad p(v_i)^{(t)} = \frac{1}{|D_v|} \sum_{d_i \in D_v} p(d_i)^{(t-1)}$$

5 **end for**

6 **for** each paper in D **do**

$$7 \quad p(d_i)^{(t-1)} = p(d_i)^{(t-1)}, \quad p(d_i)^{(t)} = \zeta p(d_i)^{(t-1)} + \frac{1 - \zeta}{2} \left(\sum_{u \in U_d} \frac{p(u_i)^{(t)}}{|U_d|} + \sum_{v \in V_d} \frac{p(v_i)^{(t)}}{|V_d|} \right)$$

8 **end for**

9 $t \leftarrow t+1$

10 **while** $t \leq M$ **do**

11 **End while**

12 **Return** $(p(d_i), p(u_i), p(v_i))$

According to the similarity calculation of the ultimate vectors of papers, we can get the paper clusters. Analogously, we also can obtain the author clusters and venue clusters.

In the previous section, each meaningful frequent phrase has been mapped to the latent semantic space with the vector $p(mfp_i)$ by LSA. Select the centroid of each paper cluster, the Euclidean distance of each meaningful frequent phrase to the centroid can be calculated by:

$$dist(p(mfp_i), p(c_j)) = \sqrt{\sum_{z=1}^k (p(mfp_i)_z - p(c_j)_z)^2} \quad (5)$$

where vector $p(c_j)$ represents the centroid of each paper cluster, $p(c_j)_z$ represents the dimensional value to the corresponding topic.

According to formula (5), we assign each meaningful frequent phrase to the paper cluster with the smallest Euclidean distance. Now each paper cluster can be treated as an optimized topic associated with both papers and meaningful frequent phrases. The topics are modeled as a set of meaningful frequent phrases.

3.4 Topic Description

Although the discovered topics are modeled as a set of meaningful frequent phrases, in order to make them more understandable, it is necessary to calculate a topic description for each topic. The topic description is expected to basically capture the meaning of the topic.

For each topic, we pick the meaningful frequent phrase with the biggest cosine similarity with the documents in that topic as a topic description. For the document $p(d_i)$ and meaningful frequent phrase $p(mfp_j)$ in each topic, cosine similarity between them can be calculated: $sim(p(mfp_j), p(d_i)) = \frac{\sum_{z=1}^k p(mfp_j)_z p(d_i)_z}{\sqrt{\sum_{z=1}^k p(mfp_j)_z^2} \sqrt{\sum_{z=1}^k p(d_i)_z^2}}$.

For each meaningful frequent phrase, we add together its cosine similarity with the documents that belong to the same topic. Obviously, topic description is the meaningful frequent phrase with the biggest cosine similarity summation in each topic.

4 Experiments

In this section, we apply LSA-PTM to the real data and show its effectiveness over the state-of-the-art models.

4.1 Dataset and Metric

We evaluate the effectiveness LSA-PTM on Digital Bibliography and Library Project (DBLP) dataset [11]. In our experiments, we use a DBLP subset that belongs to four

areas, i.e. database (DB), data mining (DM), information retrieval (IR) and artificial intelligence (AI), and contains 1200 documents (abstracts and titles), 1576 authors and 8 conferences. We extract 1660 unique meaningful frequent phrases from this data collection with threshold $\psi = 5$. A heterogeneous information network can be built on this dataset in which there are three types of objects: papers with text content, authors and venues without text content, and two types of relationships: paper-author and paper-venue, which consist of a total number of 4339 links. Note that we set the number of topics (k) to be 4.

To quantitatively identify the effectiveness of LSA-PTM, we adopt *F measure* as our metric [12]. The *F-measure* combines the *Precision* and *Recall* together. In our experiments, there are four categories. For each topic cluster, we calculate the *Precision* and *Recall* with regard to each given category. Specifically, for the obtained

$$\text{cluster } j \text{ and the given category } i : \text{Precision} (i, j) = \frac{n_{ij}}{n_i}, \text{Recall} (i, j) = \frac{n_{ij}}{n_j},$$

$$F(i, j) = \frac{2 \times \text{Precision}(i, j) \times \text{Recall}(i, j)}{\text{Precision}(i, j) + \text{Recall}(i, j)}, \text{ where } n_{ij} \text{ is the number of}$$

members of category i in cluster j , n_i is the number of members in the given category i , n_j is the number of members in cluster j and $F(i, j)$ is the *F* measure of cluster j and category i . The *F-measure* of the whole clustering results is defined as a weighted sum over all the categories as follows: $F = \sum_i \frac{n_i}{n} \max_j \{F(i, j)\}$, where the max is taken over all clusters.

4.2 Experiment Results

We first analyze the topic modeling results with case studies. Then we discuss how to set the harmonic parameter of LSA-PTM to achieve the best performance. Finally, experiments are conducted to compare the performance of object clustering with different models.

Topic Analysis and Case Study

In our model, we set the harmonic parameter $\zeta = 0.9$ and the iterations $M = 9$, the topic modeling results are shown in Table 1. Each discovered topic is modeled as a set of meaningful frequent phrases with a topic description in bold. For comparison, we conduct PLSA [5] and TMBP-Regu [7] on DBLP dataset. The most representative terms generated by PLSA and TMBP-Regu are shown in Table 2 and Table 3 respectively. Contrast of Table 1, Table 2 and Table 3, it is more easily to understand the meanings of the four topics by the topic descriptions, i.e., “database systems”, “data mining”, “information retrieval” and “artificial intelligence”, derived from LSA-PTM. For the first three topics of PLSA and TMBP-Regu, all these terms can describe the topics to some extent. For Topic 4, the topic description, “artificial intelligence”, derived from LSA-PTM is obviously more telling than “problem, algorithm, paper” derived by PLSA and “learning, based, knowledge” by TMBP-Regu. Thus, from the view of readability of the topics, LSA-PTM is better than PLSA as well as TMBP-Regu.

Table 1. The topic representation generated by LSA-PTM

Topic 1	Topic 2	Topic 3	Topic 4
database systems database system database management distributed databases relational databases database structure relational data model query processing	data mining clustering algorithms classification algorithms data cube data analysis knowledge discovery mining problems data warehouse	information retrieval language model web search retrieval performance search engine retrieval models search results semantic search	artificial intelligence knowledge-based algorithms machine learning pattern recognition knowledge engineering user interface knowledge based systems expert systems

The bold phrase in each topic is the topic description.

Table 2. The representative terms generated by PLSA

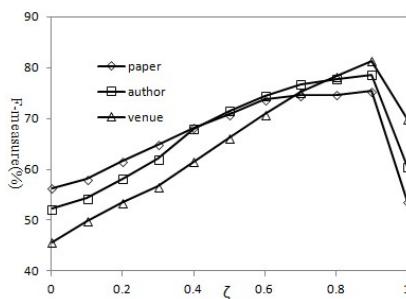
Topic 1	Topic 2	Topic 3	Topic 4
data database systems query system databases management distributed	data mining learning based clustering classification algorithm image	information retrieval web based learning knowledge text search	problem algorithm paper reasoning logic based time algorithms

Table 3. The representative terms generated by TMBP-Regu

Topic 1	Topic 2	Topic 3	Topic 4
data database query databases systems queries system processing	data mining algorithm clustering classification based algorithms rules	information web retrieval search based text language user	learning based knowledge model problem reasoning system logic

Parameter Analysis

In our method, there exists a harmonic parameter ζ , in this section, we will study and evaluate the effect of the parameter ζ .

**Fig. 3.** The effect of varying parameter ζ in LSA-PTM

As mentioned in section 3.3, the harmonic parameter is used to control the balance between the inherent topics of documents from LSA and the propagated topics. When $\zeta = 1$, it is the LSA model. Figure 3 shows the performance of LSA-PTM with the varied harmonic parameter. We can see that the performance is improved over the LSA model when incorporating topic propagation on the heterogeneous information network with $\zeta < 1$. Note that with the decrease of ζ , the performance turns worse, and even worse than the LSA model, because the model relies more on the topic consistency while ignores the intrinsic topics of the documents. According to Figure 3, we set the harmonic parameter $\zeta = 0.9$ in other experiments.

Clustering Performance Comparison

We apply our model on the task of object clustering using DBLP dataset. The hidden topics extracted by the topic modeling approaches can be regarded as clusters. We calculate *F-measure* with the provided category to evaluate the clustering results. We compare the proposed LSA-PTM with the state-of-the-art methods: latent semantic analysis (LSA) [9], probabilistic latent semantic analysis (PLSA) [5], author-topic model (ATM) [13]. For each method, 15 test runs were conducted, and the final performance scores were obtained by averaging the scores from the 15 tests. The italic results of LSA in Table 3 are obtained by LSA-PTM with $\zeta=1$.

Table 4. Clustering performance of different methods on DBLP

Metric (%)	F-measure				
	Object	Paper	Author	Venue	Average
LSA	53.59	<i>60.40</i>	69.9	61.30	
PLSA	58.45	-	-	58.45	
ATM	64.00	61.13	-	62.56	
LSA-PTM	75.35	78.61	81.36	78.44	

As shown in Table 3, the proposed LSA-PTM achieves the best overall performance. This shows that integrating heterogeneous information network with topic modeling by topic propagation, LSA-PTM can have a better topic modeling power for clustering objects.

5 Related Work

Topic modeling has attracted much attention recently in multiple types of text mining tasks, such as information retrieval [14, 15], geographical topic discovery [16], and extract scientific research topics [17, 18].

Topic modeling on heterogeneous information networks was paid little attention in existing literatures. The latest study, TMBP-Regu [7], first directly incorporates heterogeneous information network and textual documents with topic modeling. But the topic modeling results of TMBP-Regu are presented by word distributions like most of existing topic models [3, 4, 5, 6]. Although the discovered topics interpreted

by the top keywords in the word distributions are intuitively meaningful, it is still difficult for a user to accurately comprehend the meaning of each topic. In our model, the discovered topics are modeled as a set of meaningful frequent phrases accompanied with topic descriptions that successfully solve the problem of intelligibility of discovered topics.

Many topic models, such as Latent Semantic Analysis (LSA) [9], Probabilistic Latent Semantic Analysis (PLSA) [5] and Latent Dirichlet Allocation (LDA) [6], have been successfully applied to or extended to many data analysis problems, including document clustering and classification [2], author-topic modeling [13]. However, most of these models merely consider the textual documents while ignoring the network structures. Recently, several proposed topic models, such as LaplacianPLSI [2], NetPLSA [3] and iTopicmodel [4], have combined topic modeling and network structures, but they only emphasize on the homogeneous networks, such as document network and co-authorship network, but not heterogeneous information networks. Our proposed model takes the heterogeneous network structures into topic modeling by topic propagation between textual objects and other types of objects. Experimental results are desirable.

6 Conclusion and Future Work

In this paper, LSA-PTM is proposed for topic modeling heterogeneous information networks and also solves the readability of the topic modeling results. First, LSA-PTM extracts meaningful frequent phrases from documents. Then latent semantic analysis is conducted on these phrases, so as to obtain the inherent topics of the documents. Moreover, we introduce a topic propagation method that optimizes the inherent topics using heterogeneous network structures between different objects and ultimately enhances the topic modeling. To make the topics more understandable, a topic description is calculated for each topic. Experimental results show the effectiveness of LSA-PTM. Our future work will apply LSA-PTM on large scale dataset to examine and verify its efficiency.

Acknowledge. This work was supported by the National Key Technologies R&D Program (Grant No.2012BAH54F04), the National Natural Science Foundation of China (Grant No.61003051), and the Natural Science Foundation of Shandong Province of China (Grant No.ZR2010FM033).

References

1. Sun, Y., Han, J., Yan, X., Yu, P.: Mining Knowledge from Interconnected Data: A Heterogeneous Information Network Analysis Approach. In: Proceedings of the VLDB Endowment, pp. 2022–2023. ACM Press (2012)
2. Cai, D., Mei, Q., Han, J., Zhai, C.: Modeling hidden topics on document manifold. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 911–920. ACM Press, New York (2008)

3. Mei, Q., Cai, D., Zhang, D., Zhai, C.: Topic modeling with network regularization. In: Proceedings of the 17th International Conference on World Wide Web, pp. 101–110. ACM Press, New York (2008)
4. Chen, Sun, Y., Han, J., Yu: itopicmodel: Information network-integrated topic modeling. In: Proceedings of Ninth IEEE International Conference on Data Mining, pp. 493–502. IEEE Computer Society, Miami (2009)
5. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57. ACM, New York (1999)
6. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of Machine Learning Research, 993–1022 (2003)
7. Deng, H., Han, J., Zhao, B., Yu, Y.: Probabilistic topic models with biased propagation on heterogeneous information networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1271–1279. ACM Press, New York (2011)
8. Simitsis, A., Baid, A., Sismanis, Y., Reinwald, B.: Multidimensional content exploration. The VLDB Endowment, 660–671 (2008)
9. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society for Information Science and Technology, 391–407 (1990)
10. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. Numerische Mathematik, 403–420 (1970)
11. DBLP, <http://www.informatik.uni-trier.de/~ley/db/>
12. Larsen, B., Aone, C.: Fast and Effective Text Mining Using Linear-time Document Clustering. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 16–22. ACM Press, New York (1999)
13. Steyvers, M., Smyth, P., Rosen-Zvi, M., Griffiths, T.L.: Probabilistic author-topic models for information discovery. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 306–315. ACM Press, New York (2004)
14. Wei, X., Croft, W.B.: LDA-based document models for ad-hoc retrieval. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 178–185. ACM Press, New York (2006)
15. Zhai, C., Lafferty, J.: Model-based feedback in the language modeling approach to information retrieval. In: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 403–410. ACM Press, New York (2001)
16. Yin, Z., Cao, L., Han, J., Zhai, C.: Geographical topic discovery and comparison. In: Proceedings of the 20th International Conference on World Wide Web, pp. 247–256. ACM Press, New York (2011)
17. He, D., Parker, D.S.: Topic dynamics: an alternative model of bursts in streams of topics. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 443–452. ACM Press, New York (2010)
18. He, Q., Chen, B., Pei, J., Qiu, B., Mitra, P.: Detecting topic evolution in scientific literature: how can citations help? In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 957–966. ACM Press, New York (2009)

Improving Semi-supervised Text Classification by Using Wikipedia Knowledge

Zhilin Zhang, Huaizhong Lin, Pengfei Li, Huazhong Wang, and Dongming Lu

College of Computer Science and Technology, Zhejiang University,
38 Zheda Road, Hangzhou 310027, P.R. China
zhzh1989@gmail.com, {linhz,1dm}@zju.edu.cn,
pf_li@foxmail.com, wanghuazhong@yahoo.cn

Abstract. Semi-supervised text classification uses both labeled and unlabeled data to construct classifiers. The key issue is how to utilize the unlabeled data. Clustering based classification method outperforms other semi-supervised text classification algorithms. However, its achievements are still limited because the vector space model representation largely ignores the semantic relationships between words. In this paper, we propose a new approach to address this problem by using Wikipedia knowledge. We enrich document representation with Wikipedia semantic features (concepts and categories), propose a new similarity measure based on the semantic relevance between Wikipedia features, and apply this similarity measure to clustering based classification. Experiment results on several corpora show that our proposed method can effectively improve semi-supervised text classification performance.

Keywords: Semi-supervised Text Classification, Wikipedia, Clustering Based Classification.

1 Introduction

With the exponential growth of online documents in the World Wide Web, the task of assigning natural language text documents on the basis of their contents to one or more predefined categories or classes have become interesting for the text mining research community. This task is commonly referred to as text classification[18]. However, the major bottleneck of most existing text classification algorithms is that they require sufficient labeling of documents. Labeling is a task usually done manually, which is expensive, time consuming and error prone. At the same time, there are often abundant unlabeled documents that are much easier to obtain. How to effectively use unlabeled samples in text classification field has become an active research problem. The general problem of exploiting unlabeled data to improve classification accuracy leads to semi-supervised text classification.

Previous researches have come up with various methods that aim to reduce efforts for labeling tasks. Some studies proposed models that learn from labeled and unlabeled data, such as transductive support vector machines (TSVM) [13]

and Naive Bayes with EM. An alternative strategy that employs clustering methods in text classification is clustering based classification[8],[9],[10],[16]. And it has become an interesting problem for the text mining research community.

However, previous methods archived limited improvement because they represented the document as the vector of terms which appear in the document. Some literatures[1],[2],[3],[4],[5],[6],[11] employed external knowledge such as Wikipedia to enrich a document with semantic features. But the authors still adopted the cosine similarity measure when computing the similarity between documents, which ignores the semantic relationships between Wikipedia features. In this paper, we address this problem by proposing a new similarity measure considering both the term frequency and wikipedia features.

Our contributions can be summarized as follows. (1) We enrich documents by mining semantic relationships in Wikipedia and present a new document similarity measure . (2) We incorporate the knowledge obtained from Wikipedia and use our proposed similarity measure in semi-supervised text classification to achieve increased accuracy. (3) We conduct comprehensive experiments to validate our approach and study related issues. The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 introduces our proposed method of utilizing Wikipedia semantic features to improve semi-supervised text classification. In Section 4, we present and discuss the experimental results. Finally, we conclude the paper in section 5.

2 Related Work

2.1 Semi-supervised Text Classification

Recently, there is a growing amount of researches in semi-supervised text classification and different methods are proposed according to different view of unlabeled data, such as transductive support vector machines (TSVM) [13], Naive Bayes with EM[14],[15] and clustering based classification[8],[9],[10],[16].

TSVM[13] adds a constraint to the SVM optimization function in order to preserve the margin over unknown test labels. It expects to find a low-density area of data and constructs a linear separator in this area so that the margin over both the labeled data and the unlabeled data can be maximized. Nigram et al.[14] combined EM with Naive Bayes to construct a model with the unlabeled data. Their experimental results showed improved performance over supervised classifiers.

Another strategy is using clustering to boost classification [8]. In this method a clustering algorithm is applied to both the labeled and unlabeled data. Then the labels of the labeled data samples are propagated to the unlabeled samples which are closest to cluster centroids. Finally, some classification methods are used to build the classifier with the expanded training dataset. Experiment results demonstrate that this method outperforms existing algorithms such as TSVM and Naive Bayes with EM.

2.2 Utilization of Wikipedia

Nowadays, people focus on enriching document representation with semantic features exploited from Wikipedia. Literatures focus on some directions.

One is how to fully mine semantic relation between words. According to the previous researches, we can find out synonyms by redirect links, polysemous words by the disambiguation page, hypernym by hierarchical relations between concepts and categories and association relations by internal page links from Wikipedia.

As for the problem of searching candidate Wikipedia features from text documents, Gabrilovich [11] matched documents with the most relevant articles of Wikipedia by an auxiliary text classifier, and then augmented the bag of words representation with concepts represented by the relevant Wikipedia articles. Hu [1] presented two approaches, the exact match and the relatedness match, to map text documents to Wikipedia concepts and categories. Wang [3] adopted the forward maximum matching and the window filtering condition method to search candidate concepts. He then made disambiguation by using text similarity and context information.

Semantic features extracted from Wikipedia can be utilized to enhance text mining tasks, such as text classification [3],[5],[11], document clustering [1], QA [2] and online advertising [6]. However, few works have been reported about semi-supervised text classification.

3 Enrichment of Text Documents with Wikipedia Features

In this section, we first introduce Wikipedia. Then we describe the methods of searching candidate concepts in text documents. Finally, we enrich text documents with concepts and categories.

As the largest encyclopedia in the world, Wikipedia surpasses other knowledge bases due to its coverage of concepts, rich semantic knowledge and up-to-date content. The title of each article is defined as a Wikipedia concept, which is a concise, well-formed phrase and describes a single topic. Many meaningful relation structures can be mined from Wikipedia, such as synonym, polysemy, associative relation and categories .

3.1 Searching Candidate Concepts

To enrich text documents with Wikipedia knowledge, the first issue is to search the candidate concepts from the document and filter out the noise concepts to get the most promising ones. Wikipedia has such a large scale of concepts that every document phrase can be mapped to at least one article and most phrases mapped to several. To decrease meaningless mappings and deal with word ambiguities, we build a phrase index which includes the phrases of Wikipedia concepts, their synonym, polysemy and categories in the Wikipedia thesaurus. Based on the

generated Wikipedia phrases index, we use the maximum matching algorithm with the window filtering condition to search candidate concepts followed with Pu [5]. The window filtering condition means that every word of a candidate concept must appear in the sequence within a window of certain length, which guarantees all words of a concept appear in a sequence within certain distance. We adopt two strategies presented in [5] to do word sense disambiguation: the first one is to utilize text similarity for disambiguation; the second one is to disambiguate with context.

3.2 Enrichement of Text Document with Concepts

Each Wikipedia article contains a lot of hyperlinks, which express semantic relationships between concepts. Hu [4] introduced two measures to quantify the strength of associative relations between concepts: content-based s_{tf-idf} , out-link category-based s_{olc} . The measure s_{tf-idf} reflects the cosine similarity of article pairs in Wikipedia. The measure s_{olc} compares the cosine similarity between the two articles' out-linked category vectors. The vector is constructed by the categories that out-linked articles of the original article belong to. Combine the two measures linearly to compute the relatedness between concepts:

$$s^{rel}(c_i, c_j) = \lambda s_{tf-idf} + (1 - \lambda) s_{olc}. \quad (1)$$

For synonyms, Wikipedia guarantees that there is only one article for those concepts, and these equivalent concepts are grouped to the preferred concepts by redirect hyperlink. For instance, the preferred concept of U.S.A, U.S, USA and United States is United States. As Hu [1] pointed out that synonymies of a given concept cannot be ranked, and if all its synonymies are added into documents, the recall performance may be sometimes improved, but the precision of clustering will be decreased. Also, if many less related concepts be added, the precision of clustering will be reduced. In our experiment, only the preferred concept and the top 5 most relevant concepts are appended to document for each identified concept and the parameter λ is set to 0.5.

3.3 Enrichement of Text Document with Categories

In Wikipedia, both concepts and categories belong to more than one category, which form a directed acyclic graph $G = (V, E)$. $V = \{v_1, v_2, \dots, v_n\}$ is the collections of vertexes contained in G , where the vertex $v_i = \{\text{concept}|\text{category}\}$ is either a concept or a category. $E = \{e_1, e_2, \dots, e_n\}$ is the set of edges in the graph and each edge concatenates a pair of vertexes, which indicates that there is hierarchical relationship between them. For the two vertexes v_i and v_j , the less vertexes along the shortest path between the two vertexes, the more close of their semantic relatedness. For example, the distance is only one between concept and its ancestor category. The formula[19] that measures the degree of the semantic relatedness between concept and category or between categories is defined as follows:

$$s^{path}(v_i, v_j) = 1 - \frac{sl(v_i, v_j)}{2L}. \quad (2)$$

Where, $sl(v_i, v_j)$ is the number of vertexes along the shortest path between v_i and v_j , and L is the maximum depth of the graph.

A concept belongs to a few levels of category. Intuitively, those high level categories have less influence than those low level categories since low level categories are more specific and therefore can depict the articles more accurately. Lots of categories may bring some noise and reduce the precision of clustering. According the experiment results in [1], we append only the direct ancestor category to documents.

4 Improvement in Clustering Based Classification

In this section, We first describe our improvement in measuring the similarity between documents enriched with wikipedia knowledge. Then we improve the performance of clustering based classification using our proposed similarity measure.

4.1 Improvement in Similarity Measure

Traditional Similarity Measure. We define $D = \{D_l, D_u\}$ as the training dataset which has been enriched with Wikipedia features including concepts and categories. D_l is the labeled dataset, while D_u is the unlabeled dataset, T is the set of all different terms occurring in D . Firstly, stopwords are removed from T using a standard stopwords list¹. Then words in each document are stemmed using the Porter stemmer². Let $T = \{t_1, t_2, \dots, t_n\}$. Each document $d_i \in D$ can be represented as a term vector using the vector space model, means $(w(t_1, d_i), w(t_2, d_i), \dots, w(t_n, d_i))$. Where $w(t_j, d_i)$ is the weight of the term t_j in d_i which is defined as TF*IDF formula:

$$w(t_j, d_i) = TF_{ij} \times \log(|D|/DF_j). \quad (3)$$

For $d_i, d_j \in D$, their term frequency based similarity measure is defined as:

$$sim^w(d_i, d_j) = \frac{\sum_{k=1}^n w(t_k, d_i) \cdot w(t_k, d_j)}{\sqrt{\sum_{k=1}^n w^2(t_k, d_i)} \cdot \sqrt{\sum_{k=1}^n w^2(t_k, d_j)}}. \quad (4)$$

Although d_i, d_j have been enriched with Wikipedia features, the semantic relevance between Wikipedia features is ignored if the cosine similarity measure in formula (4) is used to compute the document similarity.

¹ <http://www.lextek.com/manuals/onix/stopwords1.html>

² <http://tartarus.org/~martin/PorterStemmer/>

Wikipedia Feature Based Similarity Measure. Define C_i as the set of Wikipedia concepts occurring in document d_i and Cat_i as the set of Wikipedia categories that concepts in C_i belong to, $d_i^{wiki} = \{a_p | a_p \in (C_i, Cat_i)\}$ as the set of Wikipedia features that have been added to the document d_i . The overall semantic similarity of Wikipedia features in two documents is defined as follows:

$$mes(d_i, d_j) = \sum_{a_p \in d_i^{wiki}, a_q \in d_j^{wiki}} rel(a_p, a_q). \quad (5)$$

Where, $rel(a_p, a_q)$ represents the semantic relatedness between two Wikipedia features. We define it as follows:

$$rel(a_p, a_q) = \begin{cases} 1 & , a_p = a_q \\ s^{rel}(a_p, a_q) & , a_p \neq a_q, a_p \in C_i, a_q \in C_j \\ s^{path}(a_p, a_q) & , otherwise. \end{cases} \quad (6)$$

Where, if $a_p = a_q$, $rel(a_p, a_q)$ is equal with 1, which represents the highest relatedness. If $a_p \neq a_q$ and a_p, a_q are both concepts in d_i and d_j , then formula (1) is used to calculate the semantic relatedness of the two concepts. Otherwise, either a_p or a_q is a category in the document. In this case, $rel(a_p, a_q)$ can be measured by formula (2).

Consequently, the Wikipedia feature based similarity measure of two documents $d_i, d_j \in D$ is defined as:

$$sim^{wiki}(d_i, d_j) = \frac{mes(d_i, d_j)}{\sqrt{mes(d_i, d_i) \cdot mes(d_j, d_j)}}. \quad (7)$$

Combined Similarity Measure. Finally, to measure the similarity between documents that have been enriched with Wikipedia features, we use a linear combination of the term frequency based similarity measure and the Wikipedia feature based similarity measure, which is:

$$sim(d_i, d_j) = \alpha sim^w(d_i, d_j) + (1 - \alpha) sim^{wiki}(d_i, d_j). \quad (8)$$

Where, α is used to adjust the weight of two parts. We will demonstrate the influence of parameter changes on the classification performance later in our experiment.

4.2 Clustering Based Classification Using Wikipedia Knowledge

Clustering to Expand the Labeled Dataset. Although clustering is an unsupervised task, labeled documents can be used to improve clustering accuracy. K-medoids algorithm aims to partition the observations into k sets, so as to minimize the within-cluster sum of squares. We use it as the clustering algorithm. Further, we can guide the clustering process with the labeled documents by setting the initial parameters. Followed with [8], the k value is set to the number of

classes in the labeled data and the initial centroid of each cluster is calculated by $\mu_i = \text{avg}(\sum_{\forall j \in D_l, \text{cat}(d_j)=i} d_j), i = 1 \dots c$. We use the formula (8) as the similarity measure and run k-medoids on both labeled and unlabeled data. The clustering process is terminated when the result doesn't change anymore, or just before a labeled centroid will be assigned to a wrong cluster.

After the clustering step, we propagate the labels of the labeled data samples to the unlabeled samples which are closest to cluster centroid. To avoid importing too much noise, we define a threshold δ . If unlabeled document d_j satisfies the condition: $\text{sim}(d_j, \mu_i) > \delta; d_j \in D_u, i = 1 \dots c$, then this document can be viewed as labeled data with high confidence. As a result, the scale of labeled data is expanded.

Classification with Expanded Labeled Dataset. We run TSVM to train a classifier in the expanded labeled data and remaining unlabeled data. To make a comparison, TSVM is also used in the original dataset D as a baseline.

5 Experimental Result

5.1 Dataset

The Wikipedia database is downloaded from <http://download.wikipedia.org>, which releases its database dumps periodically . We got about 832,553 articles and 29000 categories after pre-processing and filtering.

Reuters-21578 (Reuters) is the most widely used text collection for text classification. We use the ModApte³ split to form the training set and test set, where there are 7,769 training examples and 3,019 test examples. After selecting the biggest ten classes: earn, acq, money-fx, grain, crude, trade, interest, ship, wheat, and corn, we get 6,649 training examples and 2,545 test examples.

20 newsgroups data set is collected by Ken Lang⁴. This is a well-balanced data set and consists of 18828 articles spread almost evenly over 20 different categories. The version of the data set used has no duplicates, and most of the headers are removed. 10 categories are selected randomly to be used in our experiment. Each category has approximate 1000 articles.

5.2 Performance Measures

For evaluating the effectiveness of classification, we use the standard recall, precision and F1 measure. Recall is defined to be the ratio of correct assignments by the system divided by the total number of correct assignments. Precision is the ratio of correct assignments by the system divided by the total number of the system's assignments. The F1 Measure could be used to combine precision

³ <http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-text-classification-1.html>

⁴ <http://people.csail.mit.edu/jrennie/20Newsgroups/>

and recall into a unified measure. There are two ways to measure the average F1 of a binary classifier over multi categories, namely, the macro-averaging and micro-averaging. The former way is to first compute F1 for each category and then average them, while the later way is to first compute precision and recall for all categories and use them to calculate F1. Because the micro-averaging of F1 is in fact a weighted average over all classes and is more plausible for highly unevenly distributed classes, we use this measure in the following experiments.

5.3 Overall Performance

We implement three experiments on the 20newsgroup dataset. The results are shown in Fig. 1. We first run the clustering based classification algorithm on the original dataset as the baseline. After enriching the dataset, we choose the traditional similarity measure as shown in formula (3) in the clustering step. Finally, we use our proposed similarity measure as shown in formula (8) in the clustering step. The results are shown in Fig. 1 as "BASE", "Tr_W" and "New_W" respectively.

As can be seen from Fig. 1, our proposed algorithm performs better on the dataset with Wikipedia knowledge than on the original dataset, so do our proposed similarity measure than the traditional measure especially when the number of the labeled data is very small. For example, when the number of the labeled data in all training dataset is 40, our proposed algorithm achieve 7.5% improvement.

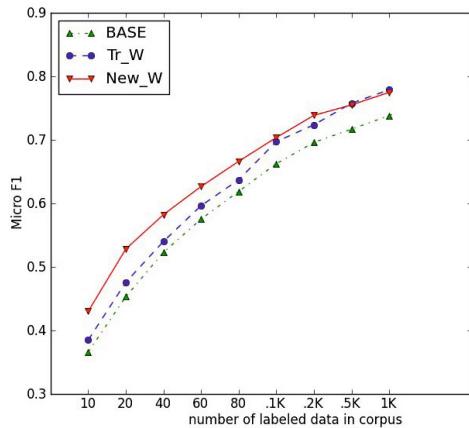


Fig. 1. Comparison of clustering based classification performances on 20newsgroup dataset. 1) traditional similarity measure as baseline; 2) traditional similarity measure in the clustering step after enriching the dataset with Wikipedia knowledge; 3) new similarity measure in the clustering step after enriching the dataset with Wikipedia knowledge.

Similar results can be found on Reuters-21578 as shown in Fig. 2. By using Wikipedia knowledge, our proposed algorithm still outperforms when the number of the labeled data is very small.

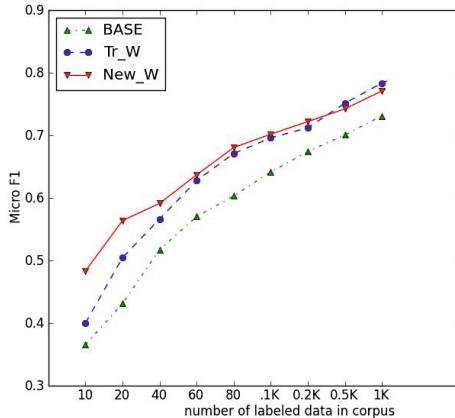


Fig. 2. Comparison of clustering based classification performances on Reuters-21578 dataset. 1) traditional similarity measure as baseline; 2) traditional similarity measure in the clustering step after enriching the dataset with Wikipedia knowledge; 3) new similarity measure in the clustering step after enriching the dataset with Wikipedia knowledge.

Thus from Fig. 1 and Fig. 2, it is possible to conclude that when using Wikipedia knowledge, the algorithm performs better, especially when the number of labeled sample is small. Another conclusion can be made that our proposed similarity measure can help the clustering algorithm to find more appropriate labels for the unlabeled documents, then use these to expand the training dataset.

However, with the increasing number of labeled data, the performance of clustering based classification grows slowly, and sometimes drops. This is because when the number of labeled samples exceed a certain range, the original training dataset is sufficient for classification and expanding training dataset with unlabeled samples in the clustering step may introduce noise.

5.4 Clustering Performance

We evaluate the clustering performance by the purity measure which is based on precision measure in information retrieval field. We run k-medoids algorithm with the traditional similarity as the baseline. Then the combined similarity measure is implemented instead. Table 1 shows the clustering performance on the Reuters-21578 and 20newsgroup.

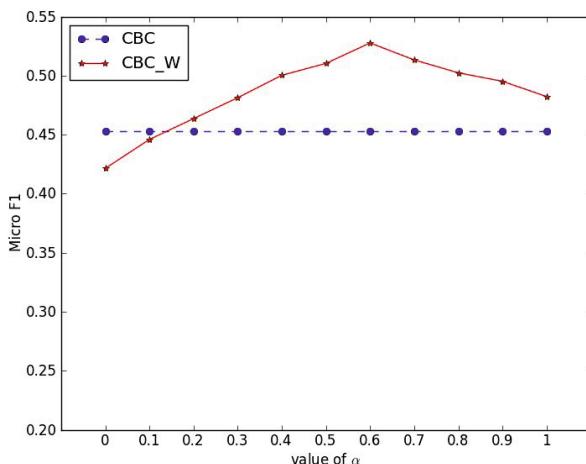
Table 1. Comparison of the performance clustering with different similarity measure on the original Reuters-21578 and 20newsgroup dataset

Labeled docs	Reuters-21578		20newsgroup	
	BASE	CBCW	BASE	CBCW
10	0.934	0.951	0.853	0.901
20	0.904	0.92	0.828	0.873
40	0.873	0.896	0.796	0.832
60	0.842	0.869	0.786	0.801
80	0.837	0.836	0.781	0.803
100	0.826	0.832	0.779	0.796
200	0.844	0.856	0.785	0.795
500	0.835	0.839	0.781	0.788
1000	0.865	0.869	0.789	0.786

From Table1, we can see that our improved similarity measure performs better than the traditional similarity measure. The precision of the clustering is very high when the number of labeled documents is relatively less, since the number of the most confident unlabeled documents which are viewed as labeled documents is small. As the number of labeled documents becomes larger, the precision will be reduced.

5.5 Parameter Tuning

We then discuss the relationship between the parameter and the classification performance. To observe the significant improvement in performance, we run this experiment when the number of labeled samples is small.

**Fig. 3.** The F1 value of classification with different α on 20newsgroup when the number of labeled documents is 40

The value of α is set to 0.1, 0.2, ..., 1.0 respectively to tuned the similarity measure, and various classifier are build. Then we observe the classification performance in the test dataset. Fig. 3 shows the performance with various values of α on 20newsgroup collection.

We can see that the best performance appears when α is near 0.6. Out of our expectation, when α is very small ,for example $\alpha = 0$, the performance is worse than the baseline which demonstrates that Wikipedia features cannot completely take place of the term vector to represent the document.

6 Conclusion and Future Work

This paper presents a method to improve semi-supervised text classification performance by using Wikipedia semantic knowledge. First, we map text documents to Wikipedia concepts, make word sense disambiguation to overlapping ambiguity and polysemous ambiguity, and enrich documents with Wikipedia features (synonyms, hyponyms and associative concepts). Then Wikipedia feature based similarity measure is proposed to compute the semantic similarity of added Wikipedia features in documents. By combining term frequency based similarity measure and Wikipedia feature based similarity measure in the clustering step, we can label more appropriate unlabeled documents to expand training dataset. Experiment results show that this new method can boost classification performance even when the number of labeled samples is small. For future works, more information in Wikipedia can be minded, as Wikipedia contains so much information, and our method only explores part of them. To evaluate the wide suitability and effectiveness of our new proposed similarity measure, more experiments need to be done.

Acknowledgments. This work was supported in part by the national science and technology support program grant 2012BAH03F02 and 2013BAH62F02, the Public Technology Research Program of Zhejiang Province grant 2010C33151, Science and technology innovation team of Zhejiang Province grant 2010R50040, the cultural relic protection science and technology project of Zhejiang Province.

References

1. Hu, X., Zhang, X., et al.: Exploiting Wikipedia as external knowledge for document clustering. In: ACM SIGKDD, pp. 389–396. ACM, New York (2009)
2. Cai, L., Zhou, G., et al.: Large-scale question classification in cQA by leveraging Wikipedia semantic knowledge. In: Proceedings of ACM CIKM, pp. 1321–1330. ACM, New York (2011)
3. Wang, P., Domeniconi, C.: Building semantic kernels for text classification using wikipedia. In: Proceedings of ACM SIGKDD, pp. 713–721. ACM, New York (2008)
4. Hu, J., Fang, L., Cao, Y., Zeng, H.-J., Li, H., Yang, Q., Chen, Z.: Enhancing text clustering by leveraging Wikipedia semantics. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 179–186. ACM, New York (2008)

5. Pu, W., Jian, H., et al.: Improving Text Classification by Using Encyclopedia Knowledge. In: Proceedings of 7th IEEE ICDM, pp. 332–341. IEEE Press, New York (2007)
6. Wu, Z., Xu, G., et al.: Leveraging Wikipedia concept and category information to enhance contextual advertising. In: Proceedings of ACM CIKM, pp. 2105–2108. ACM, New York (2011)
7. Banerjee, S.: Improving text classification accuracy using topic modeling over an additional corpus. In: Proceedings of ACM SIGIR, pp. 867–868. ACM, New York (2008)
8. Hua-Jun, Z., Xuan-Hui, W., et al.: CBC: clustering based text classification requiring minimal labeled data. In: 3rd IEEE International Conference on ICDM, pp. 443–450. IEEE Press, New York (2003)
9. Dai, W., Xue, G.R., et al.: Co-clustering based classification for out-of-domain documents. In: Proceedings of ACM SIGKDD, pp. 210–219. ACM, New York (2007)
10. Kyriakopoulou, A., Kalamboukis, T.: Using clustering to enhance text classification. In: Proceedings of ACM SIGIR, pp. 805–806. ACM, New York (2007)
11. Gabrilovich, E., Markovitch, S.: Overcoming the brittleness bottleneck using wikipedia: enhancing text categorization with encyclopedic knowledge. In: Proceedings of AAAI, pp. 1301–1306. AAAI Press (2006)
12. Ko, Y., Seo, J.: Text classification from unlabeled documents with bootstrapping and feature projection techniques. Information Processing & Management, 70–83 (2009)
13. Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. In: Proceedings of the 16th International Conference on Machine Learning, pp. 200–209. Morgan Kaufmann Publishers Inc. (1999)
14. Nigam, K., McCallum, A.K., et al.: Text classification from labeled and unlabeled documents using EM. In: Machine Learning, pp. 103–134 (2000)
15. Su, J., Shirab, J.S., Matwin, S.: Large scale text classification using semi-supervised multinomial naive bayes. In: ICML, New York, NY, USA, pp. 25–32 (2011)
16. Nizamani, S., Memon, N., et al.: CCM: A Text Classification Model by Clustering. In: Advances in Social Networks Analysis and Mining (ASONAM), pp. 461–467. IEEE Press, New York (2011)
17. Zhao, Y., Karypis, G.: Criterion functions for document clustering: Experiments and analysis. Technical Report, Department of Computer Science, University of Minnesota (2002)
18. Vogrinčić, S., Bosnić, Z.: Ontology-based multi-label classification of economic articles. Computer Science and Information Systems, 101–119 (2011)
19. Strube, M., Ponzetto, S.P.: WikiRelate! Computing semantic relatedness using Wikipedia. In: Proceedings of AAAI. AAAI Press (2006)

Time Series Representation: A Random Shifting Perspective

Xue Bai¹, Yun Xiong¹, Yangyong Zhu¹, and Hengshu Zhu²

¹ Fudan University

² University of Science and Technology of China

{xuebai, yunx, yyzhu}@fudan.edu.cn, zhs@mail.ustc.edu.cn

Abstract. A long standing challenge for time series analysis is to develop representation techniques for dimension reduction while still preserving their fundamental features. As an effective representation technique, Symbolic Aggregate Approximation (SAX) has been widely used for dimension reduction in time series analysis. However, SAX always maps time series data into symbols by definite breakpoints. As a result, the similar points close to the breakpoints cannot be well represented, and thus lead to poor Tightness of Lower Bounds (TLB). To fill this crucial void, in this paper, we develop a time series representation method, named *Random Shifting based SAX* (rSAX), which has the ability in significantly improving the TLB of representations without increasing the corresponding granularity of representations. Specifically, the key idea of rSAX is to generate a group of breakpoints by random shifting rather than definite breakpoints. Therefore, the points close to each other will have higher probabilities to be mapped into the same symbols, while the points far away from each other will have higher probabilities to be mapped into different symbols. In addition, we also theoretically prove that rSAX can achieve better mapping performances and TLB than SAX. Finally, extensive experiments on several real-world data sets clearly validate the effectiveness and efficiency of the rSAX approach.

1 Introduction

Recent years have witnessed the increased interests in time series analysis, such as stock market analytics [1], DNA sequence mining [2], and sensor based activity recognition [3]. Time series are essentially high dimensional data [4], which impose a significant challenge on time series analysis due to the curse of dimensionality. Therefore, it is urgently important to develop representation techniques which can reduce the dimensionality of time series while still reserving their fundamental features [5]. In the literature, a number of representation techniques have been developed for time series data, such as Symbolic Aggregate Approximation (SAX) [6], Discrete Fourier Transformation (DFT) [7], and Discrete Wavelet Transformation (DTW) [8]. Among them, SAX is one of the most popular representation techniques, which has proven its effectiveness in dimensionality reduction for time series data [9,10,11,12].

Specifically, SAX aims to transform the real valued time series into some predefined symbols with the lower bounds property, which guarantees the distances between representations to be less than or equal to their true Euclidean distances of the original

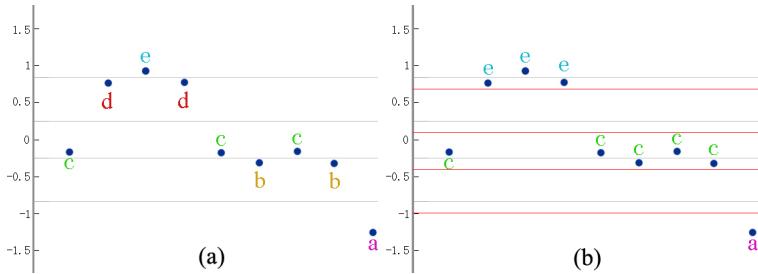


Fig. 1. The symbolic representations of points in a time series. (a) representation by the original breakpoints of SAX. (b) representation when shifting breakpoints by a small gap.

data. The main idea of SAX is to first aggregate the original time series by Piecewise Aggregate Approximation (PAA) [13], and then map them to different symbols by comparing their distances with breakpoints [6]. However, SAX always maps time series into symbols by definite breakpoints. As a result, the similar points close to the breakpoints cannot be well represented. For example, Fig. 1 (a) shows the symbolic representation of a time series by SAX with the alphabet size $\Psi = 5$ and the grey lines represent the corresponding four breakpoints. In this figure, we can observe that the $2^{th} - 4^{th}$ points are mapped to different symbols although they are actually close to each other in terms of Euclidian distances. Similarly, points $5^{th} - 8^{th}$ encounter the same mapping problem. Indeed, the reason is that the breakpoints of SAX are determined by definite values drawn from the Gaussian distribution. Therefore, if the data points are located near the breakpoints, they may be mapped to different symbols while they are close to each other. In addition, the near points close to the breakpoints can also negatively impact on the Tightness of Lower Bound (TLB) in data representation. Intuitively, to achieve better mapping results and TLB, we may have to increase the alphabet size Ψ . However, this will in turn enlarge the granularity, which may lead to the over-fitting problem and exponentially increase the candidate combinations of symbols in the subsequent time series analysis.

To address the above challenges, in this paper, we develop a time series representation approach, named *Random Shifting SAX* (rSAX), which can significantly improve the TLB of representations without increasing the corresponding granularity of representations. The key idea of rSAX is to generate a group of breakpoints by random shifting rather than definite breakpoints. Specifically, we generate “soft-borders” instead of “hard-borders” through random shifting breakpoints with small distances for several times. As a result, the points close to each other will have higher probabilities to be mapped to the same symbols. In contrast, the points far away from each other will have higher probabilities to be mapped to different symbols. For example, Fig. 1 (b) shows the symbolic representation of the same time series by rSAX, which shifts breakpoints a small distance from the original positions. We can observe that the $2^{th} - 4^{th}$ points are all represented by the symbol e and the $5^{th} - 8^{th}$ points are all represented by the symbol c . As a highlight, the contributions of this paper are summarized as follows.

Table 1. Mathematical notations

Notation	Description
$D = T_1, T_2, \dots, T_m$	The set of time series data
$T = t_1, t_2, \dots, t_n$	A time series
$S = t_p, \dots, t_{p+k-1}$	A subsequence of a time series T
$\bar{S} = \bar{s}_1, \dots, \bar{s}_w$	A Piecewise Aggregate Approximation of a subsequence S
$\hat{S} = \hat{s}_1, \dots, \hat{s}_w$	A symbol representation (word) of a subsequence S
$B = \beta_1, \dots, \beta_{\Psi-1}$	Breakpoints
w	The number of PAA elements
Ψ	Alphabet size. The total number of different symbols

- First, we develop the rSAX method for the symbolic representation of time series. rSAX allows to represent near points close to break points more effectively.
- Second, we theoretically prove that rSAX has the lower bounds property, and can achieve better TLB than SAX.
- Finally, we provide extensive experiments on real-world data sets. Experimental results clearly validate the effectiveness and efficiency of the rSAX approach.

2 Related Work

Numerous techniques have been proposed in the literature for time series representation [6,7,13]. Indeed, symbolic representation is one of the most important categories of time series representations. It has the benefit of allowing the use of many well-defined data structures and indices (e.g., suffix trees, hashing, Markov models, etc), and potentially allows the application of text-based retrieval techniques into the similarity analysis of time series. By transforming time series data to discrete symbolic words, symbolic representation can also adopt many classic statistical methods, such as the statistical tests to detect interesting subsequences of time series.

Many methods have been proposed in symbolic representations. For example, Huang *et al.* [14] proposed a method IMPACTS for time series query processing with various constraints, which is based on the suffix tree indexing. V. Megalooikonomou *et al.* [15] proposed a symbolic representation algorithm Multiresolution Vector Quantized (MVQ) with a new distance function, which keeps both local and global information about the original time series in a hierarchical mechanism.

SAX [6] is one of the most popular methods for time series symbolic representations, which has proven its effectiveness in dimensionality reduction. Thus, SAX is widely used in many applications [9,16], and many inherited methods have been further proposed in the literature. For example, iSAX [17] was proposed to enable SAX to scale to the massive data sets. Also, iSAX 2.0 [12] was further proposed for indexing and mining large-scale time series.

However, SAX and other SAX based representation approaches always map the time series into symbols by definite breakpoints, which cannot work well for representing similar points close to the breakpoints and will result worse tightness of lower bounds. In order to improve this situation and reduce the deviation caused by definite breakpoints, in this paper, we propose a random shifting process along with SAX to generate “soft-borders” instead of “hard-borders” for breakpoints, thus achieve better mapping and tighter lower bounds without increasing the alphabet size.

3 Preliminaries

In this section, we introduce several related notions and definitions which are necessary for presenting our approach. To facilitate understanding, the related mathematical notations are illustrated in Table 1.

To be specific, we define a **time series** $T = t_1, t_2, \dots, t_n$ is an ordered set of n real-valued variables, where data points t_1, t_2, \dots, t_n are temporally ordered and spaced at equal time intervals. Actually, each time series records the observations of an object over time. Then, we define a **time series data set** D as a set of m unordered time series. Moreover, a **subsequence** S of a time series $T = t_1, t_2, \dots, t_n$ is a sampling of length $k \leq n$ of contiguous position from T , i.e., $S = t_p, \dots, t_{p+k-1}$ for $1 \leq p \leq n - k + 1$.

To reduce the dimension of the original time series, the Piecewise Aggregate Approximation (PAA) [13] is applied in both SAX and our approach rSAX for creating an aggregated discrete version of time series, which can be denoted by a vector $\bar{T} = \bar{t}_1, \dots, \bar{t}_w$ (The PAA for a subsequence S is correspondingly $\bar{S} = \bar{s}_1, \dots, \bar{s}_w$). Specifically, the i^{th} element of \bar{T} is calculated by the following equation,

$$\bar{t}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} t_j.$$

Also, both SAX and our approach rSAX request a predefined alphabet size Ψ , which is leveraged for determining breakpoints of symbolic representation. To be specific, the breakpoint can be formally defined as follows [6],

Definition 1 (Breakpoints). *Breakpoints are a sorted list of numbers $B = \beta_1, \dots, \beta_{\Psi-1}$ such that the area under a $N(0, 1)$ Gaussian curve from β_i to $\beta_{i+1} = \frac{1}{\Psi}$ (β_0 and β_{Ψ} are defined as $-\infty$ and ∞ , respectively).*

For example, Fig. 2 shows the breakpoints with $\Psi = 3$. The breakpoints β_1 and β_2 divide the area under the $N(0, 1)$ Gaussian curve into three equal sub-areas, namely area a , area b , and area c . Intuitively, the points in a specific area will be represented by corresponding symbol.

Once the breakpoints are determined, a subsequence can be mapped to symbolic representation which is defined as a word. To be specific, the formal definition of word can be presented as follows [6],

Definition 2 (Word). *A subsequence S of length k can be represented as a word $\hat{S} = \hat{s}_1, \dots, \hat{s}_w$ as follows. Let α_i denote the i^{th} element of the alphabet, e.g., $\alpha_1 = \mathbf{a}$ and $\alpha_2 = \mathbf{b}$. Then the mapping from a PAA approximation \bar{S} to a word \hat{S} is obtained as follows,*

$$\hat{s}_i = \alpha_i, \text{ iff. } \beta_{j-1} < \bar{s}_i \leq \beta_j. \quad (1)$$

Moreover, SAX define a MINDIST function that returns the minimum distance between the original time series of two words $\hat{Q} = \{\hat{q}_1, \dots, \hat{q}_w\}$ and $\hat{C} = \{\hat{c}_1, \dots, \hat{c}_w\}$, which can be calculated as follows,

$$MINDIST(\hat{Q}, \hat{C}) \equiv \sqrt{\frac{n}{w}} \times \sqrt{\sum_{i=1}^w (dist(\hat{q}_i, \hat{c}_i))^2}, \quad (2)$$

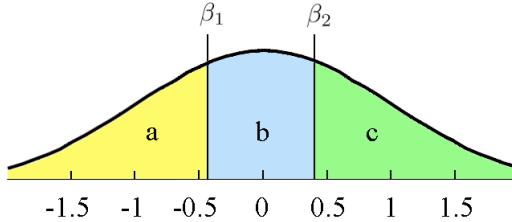


Fig. 2. The example of breakpoints with the alphabet size $\Psi = 3$

where n is the length of original time series and w is the number of PAA elements. The $dist()$ function is defined in [6], that is, the $dist()$ between two characters adjacent in alphabet is 0(e.g., $dist(a,b) = 0$), and the $dist()$ between two far-away characters is equals to the biggest Euclidean distance of the breakpoints between them(e.g., $dist(a,c) = 0.86$, $\Psi = 3$). Based on the MINDIST, a metric can be defined to measure the tightness of lower bonds by comparing the lower bound distance to the real Euclidian distance as follows,

$$TLB = \frac{MINDIST(\hat{Q}, \hat{C})}{Euclidean(Q, C)}. \quad (3)$$

TLB is a very meaningful criteria measure for time series representations, and is used to compare the performances of each representation method [5].

4 Random Shifting for Representation

In this section, we first present the details of our rSAX method and then theoretically prove that rSAX can achieve better mapping performances and TLB than that of SAX.

4.1 The Main Idea

When designing representation methods, two ideal conditions are preferred: 1) perfect mapping, which expects the similar data points are mapped to the same character; 2) tight bounding, which expects the approximate lower bounds distance is close to Euclidean distance. SAX [6] is a popular method for time series representation. However, if the pair-wise data points happen to locate in the two sides of a breakpoint, they will be represented by different symbols no matter how close they are.

To overcome above challenge, we propose a novel method rSAX that represents time series based on random shifting strategy. The main idea is to create “soft-borders” instead of “hard-borders” through random shifting breakpoints with a small distances, thus near points will have higher probabilities to be represented by same symbols, and make the bound tighter without increasing the alphabet size. To be specific, the Pseudocode of the rSAX is shown in Algorithm 1.

For a data point (or a PAA approximation \bar{C}_j) $t_j \in T, 1 \leq j \leq |T|$. Let α_k denotes the k^{th} element of the alphabet with the total size Ψ , the SAX representation of t_j is α_k

Algorithm 1. rSAX(D, Ψ, τ, w)

Input: D : the n-dimensional time series dataset; Ψ : the alphabet size; τ : the sampling times; w : PAA length.
Output: A matrix of rSAX representations M .

- 1: Initialize a list of breakpoints-lists $\beta = \beta^1, \dots, \beta^\tau$;
- 2: Look up a list of breakpoints $\beta^1 = \beta_1^1, \dots, \beta_{\Psi-1}^1$ in a statistical table;
- 3: Let d be the minimum distance of all pairwise breakpoints in β^1 ;
- 4: **for** $k = 2$ to τ **do**
- 5: Generate a random variable l_k from $(-\frac{1}{2}d, \frac{1}{2}d)$
- 6: **for** $j = 1$ to $\Psi - 1$ **do**
- 7: $\beta_j^k = \beta_j^1 + l_k$
- 8: **end for**
- 9: **end for**
- 10: **for** each time series $T_i \in D$ **do**
- 11: $P_i = PAA(T_i, w)$
- 12: **end for**
- 13: **for** each aggregated time series P_i **do**
- 14: **for** each timestamp $t_j \in P_i$ **do**
- 15: **for** $k = 1$ to τ **do**
- 16: Determine the symbol of $M[i][j][k]$ by comparing the t_j with β^j ;
- 17: **end for**
- 18: **end for**
- 19: **end for**
- 20: **return** M ;

iff. $\beta_{k-1} \leq t_j < \beta_k$ [6]. Let d denotes the minimum distance of all pairwise breakpoints, and l_i be a random variable generated by uniform distribution from $(-\frac{1}{2}d, \frac{1}{2}d)$, the breakpoints after a random shifting is changed to $\beta_k = \beta_k + l_i$. Instead of using a list of consistent breakpoints to separate data points, we generate $\tau - 1$ stochastic disturbances and add each $l_i \in \{l_1, \dots, l_{\tau-1}\}$ to initial breakpoints. The original time series are then mapped to symbolic characters for τ times, that is, there are τ versions of rSAX representations for the time series data set.

Fig. 3 shows an example of the rSAX. The alphabet size $\Psi = 4$, and the breakpoints are $B = \{-0.67, 0, 0.67\}$. Fig. 3 (a) is the result of standard SAX. The time series with PAA approximation [6] ($T = \overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4$) is mapped to the word **cbbc**. We observe that \overline{C}_1 and \overline{C}_2 are very close to each other, but happen to be differentiated by the breakpoint 0; \overline{C}_3 and \overline{C}_4 are far away from each other, however the $MINDIST(\overline{C}_3, \overline{C}_4) = MINDIST(\mathbf{b}, \mathbf{c}) = 0$ [6]. Fig. 3 (b) shows one shifting result. Let l_i be a random variable generated from $(-0.335, 0.335)$, the breakpoints are shifted to $B = \{-0.67 + l_i, 0 + l_i, 0.67 + l_i\}$. In this shifting, \overline{C}_1 and \overline{C}_2 are mapped to the same character **b**, and $MINDIST(\overline{C}_3, \overline{C}_4) = MINDIST(\mathbf{a}, \mathbf{c}) = 0.67$. Fig. 3 (c) shows another shifting result with the shifting length l_{i+1} . \overline{C}_1 and \overline{C}_2 are mapped to the same character **c** too, and $MINDIST(\overline{C}_3, \overline{C}_4) = MINDIST(\mathbf{b}, \mathbf{d}) = 0.67$.

4.2 The Mapping Property

A data point t_j changes its representation if it satisfies either of the following conditions, 1) $\beta_k + l_i \leq t_j < \beta_k$, where $-\frac{1}{2}d < l_i < 0$; 2) $\beta_{k-1} \leq t_j < \beta_{k-1} + l_i$, where $0 < l_i < \frac{1}{2}d$. Fig. 4 (a) shows an example of changeable intervals for a fixed $l_i = 0.1$. The data points within the yellow intervals change their representations from α_k to α_{k-1} , while the data points in the grey intervals remain the same. Fig. 4 (b) shows the

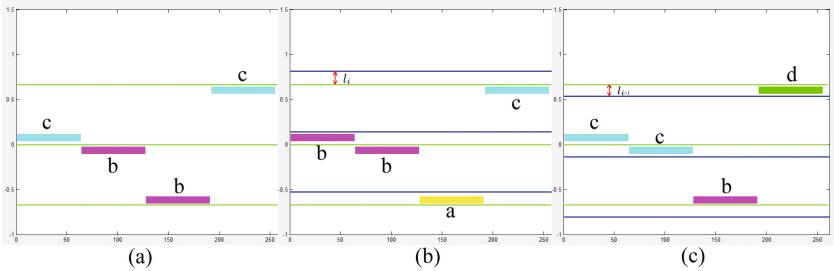


Fig. 3. The representation results of rSAX. (a) The standard SAX segmentation. (b) The breakpoints shift l_i from their initial positions. (c) The breakpoints shift l_{i+1} from their initial positions.

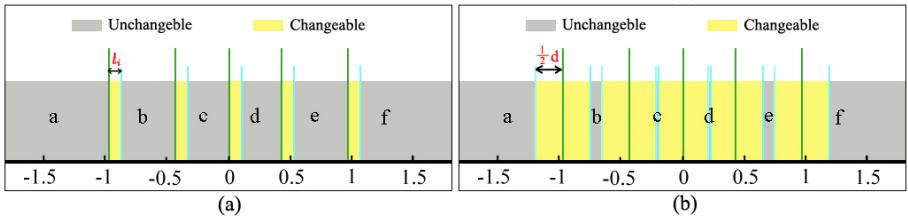


Fig. 4. The unchangeable and changeable intervals of rSAX (alphabet size $\Psi = 6$). (a) The changeable intervals (yellow) with a l_i generated from $(-\frac{1}{2}d, \frac{1}{2}d)$. (b) With a fixed d , the yellow areas show all possible changeable intervals while the grey ones never change.

total possible changeable intervals (yellow) during τ times of random shifting. Data points fell in the grey intervals never change their representations.

For a data point t_j , if $\frac{\beta_{k-1} + \beta_k}{2} \leq t_j < \beta_k$, it may be shifted from α_k to α_{k+1} . Let p_{change} denotes the probability that t_j represented by α_{k+1} , $p_{change} = \frac{t_j + \frac{1}{2}d - \beta_k}{d}$, where d is the minimum distance of all pairwise adjacent breakpoints. If $\beta_k - \beta_{k-1} = d$, t_j always has a chance to be represented by α_{k+1} . As $\frac{\beta_{k-1} + \beta_k}{2} \leq t_j < \beta_k$, $\max(p_{change}) = \frac{1}{2}$, $\min(p_{change}) = \frac{\beta_{k-1} - \beta_k}{2d} + \frac{1}{2} \geq -\frac{1}{2} + \frac{1}{2} = 0$. Thus $0 \leq p_{change} < \frac{1}{2}$.

If $\beta_k \leq t_j < \frac{\beta_{k-1} + \beta_k}{2}$, t_j may be represented by α_k or α_{k-1} during the process of random shifting. In the same way, we can calculate the probability that t_j represented by α_{k-1} : $p_{change} = \frac{\beta_{k-1} - t_j + \frac{1}{2}d}{d}$, and $0 \leq p_{change} < \frac{1}{2}$.

Theorem 1. Let t_j and $t_{j+1} = t_j + \epsilon$ denote two data points in the data set, $0 < \epsilon < d$. For $0 < \epsilon \leq \frac{d}{2}$, the upper bound of the probability that t_j and t_{j+1} are represented by the same character with the frequency no more than $\frac{1}{2}$ in the τ samples is $\exp(-\frac{(d-2\epsilon)^2\tau}{8d(d-\epsilon)})$. For $\frac{d}{2} < \epsilon < d$, the upper bound of the probability that t_j and t_{j+1} are represented by the same character with the frequency no less than $\frac{1}{2}$ in the τ samples is $\exp(-\frac{(2\epsilon-d)^2\tau}{12d(d-\epsilon)})$.

Proof. Let X_1, \dots, X_τ be independent Bernoulli trials. $Pr[X_i = 1] = p_{same} = \frac{d-\epsilon}{d}$, denotes that t_j and t_{j+1} are represented by the same character in one shifting. $Pr[X_i =$

$0] = p_{dif} = \frac{\epsilon}{d}$, denotes that the SAX representation of t_j and t_{j+1} are different in one shifting ($1 \leq i \leq \tau$). Let $X = \sum_{i=1}^{\tau} X_i$ denote the number of samples that have the same SAX representations. X follows the binomial distribution. $E[X] = \sum_{i=1}^{\tau} p_{same} = \tau \cdot p_{same}$. Using the Chernoff bound [18], for all $0 < \delta \leq 1$,

$$Pr[X \leq (1 - \delta)E[X]] \leq \exp\left(-\frac{\delta^2 E[X]}{2}\right), \quad (4)$$

$$Pr[X \geq (1 + \delta)E[X]] \leq \exp\left(-\frac{\delta^2 E[X]}{3}\right), \quad (5)$$

Let f_{same}^s be the frequency that t_j and t_{j+1} are represented by the same character in the sample, and $C^s = f_{same}^s \cdot \tau$ denote the count of the same mappings in the sample. Note that $E[C^s] = p_{same} \cdot \tau$, it is easy to see that,

$$Pr[f_{same}^s \leq (1 - \delta)p_{same}] = Pr[C^s \leq (1 - \delta)E[C^s]]. \quad (6)$$

Using the Chernoff bound in Equation 4, for all $0 < \delta \leq 1$ we have,

$$Pr[f_{same}^s \leq (1 - \delta)p_{same}] \leq \exp\left(-\frac{\delta^2 \cdot p_{same} \cdot \tau}{2}\right). \quad (7)$$

Similarly, for all $0 < \delta < 1$ we have

$$Pr[f_{same}^s \geq (1 + \delta)p_{same}] \leq \exp\left(-\frac{\delta^2 \cdot p_{same} \cdot \tau}{3}\right). \quad (8)$$

For $0 < \epsilon \leq \frac{d}{2}$, $\frac{1}{2} \leq p_{same} = \frac{d-\epsilon}{d} < 1$. By invoking Equation 6 with $\delta = \frac{2p_{same}-1}{2p_{same}} = \frac{d-2\epsilon}{2(d-\epsilon)}$, we find that

$$Pr[f_{same}^s \leq \frac{1}{2}] \leq \exp\left[-\frac{(d-2\epsilon)^2 \tau}{8d(d-\epsilon)}\right]. \quad (9)$$

For $\frac{d}{2} < \epsilon < d$, $0 < p_{same} = \frac{d-\epsilon}{d} < \frac{1}{2}$. By invoking Equation 8 with $\delta = \frac{1-2p_{same}}{2p_{same}} = \frac{2\epsilon-d}{2(d-\epsilon)}$, we find that

$$Pr[f_{same}^s \geq \frac{1}{2}] \leq \exp\left[-\frac{(2\epsilon-d)^2 \tau}{12d(d-\epsilon)}\right]. \quad (10)$$

□

The upper bound of the probability will be smaller if we increase τ . It is easy to expand the pair-wise mapping to all the objects. That is, with a bigger τ , the probability that similar objects are mapped to same symbols will be greater.

Chernoff bound describes the tail distribution of the random variables, and bounds the probability that the random variable deviates far from its expectation. Let c denote this probability. Then $1 - \delta$ describes the accuracy of the samples, and $1 - c$ describes the confidence of the samples. Then we find

$$\tau = -\frac{2\ln c}{\delta^2 p_{same}} \quad (11)$$

The confidence of the samples can be improved by increasing τ , which means the approximate similarity (represented by same symbols or not) of samples well represent that of the entire population.

4.3 The Lower Bounds Property

Similar to the *MINDIST* [6] of SAX, rSAX also has a lower bounds distance measure. The minimum distance of rSAX is defined as follows,

Definition 3 (rMinDist). *The minimum distance ($rMinDist$) of objects \hat{Q}, \hat{C} under rSAX representation is the sum of all the maximum distance of τ version of SAX representations.*

$$rMinDist(\hat{Q}, \hat{C}) \equiv \sqrt{\frac{n}{w}} \times \sqrt{\sum_{i=1}^w (rdist(\hat{q}_i, \hat{c}_i))^2} \quad (12)$$

where $rdist(\hat{q}_i, \hat{c}_i) = \text{Max}\{dist(\hat{q}_i^1, \hat{c}_i^1), \dots, dist(\hat{q}_i^\tau, \hat{c}_i^\tau)\}$.

The process of rSAX includes one time of SAX and $\tau - 1$ times of random shifting processes. For any data points q_i and c_i , rSAX selected the maximum value of all the τ distances to be the tightest bounding of their minimum distance. It is easy to see

$$\text{Euclidean}(q_i, c_i) \geq rdist(\hat{q}_i, \hat{c}_i) \geq dist(\hat{q}_i, \hat{c}_i)$$

Thus, the tightness of lower bounds (TLB) of rSAX is

$$TLB = \frac{rMinDist(\hat{Q}, \hat{C})}{\text{Euclidean}(Q, C)} \geq \frac{MINDIST(\hat{Q}, \hat{C})}{\text{Euclidean}(Q, C)} \quad (13)$$

4.4 Runtime Complexity

Let m be the number of time series, n be the length of one time series, Ψ be the alphabet size, and w be the number of timestamps for one time series after PAA. The runtime complexity for rSAX is $O(m \cdot n) + O(m \cdot w \cdot \tau)$, where τ is the shifting times.

5 Experiments

In this section, we present extensively experiments to evaluate rSAX. Specifically, all experiments were implemented by JAVA with a 2GB RAM, 2.26GHz CPU PC.

5.1 Evaluation of the Tightness of Lower Bounds

TLB is a very meaningful measure to compare the performances of representation methods [5]. The value range of TLB is from 0 (i.e., worst) to 1 (i.e., best). Here, we use the Equation 3 and Equation 13 to calculate TLB of SAX and rSAX with several real-world data sets [19]. Specifically, we randomly sampled Q and C (with replacement)

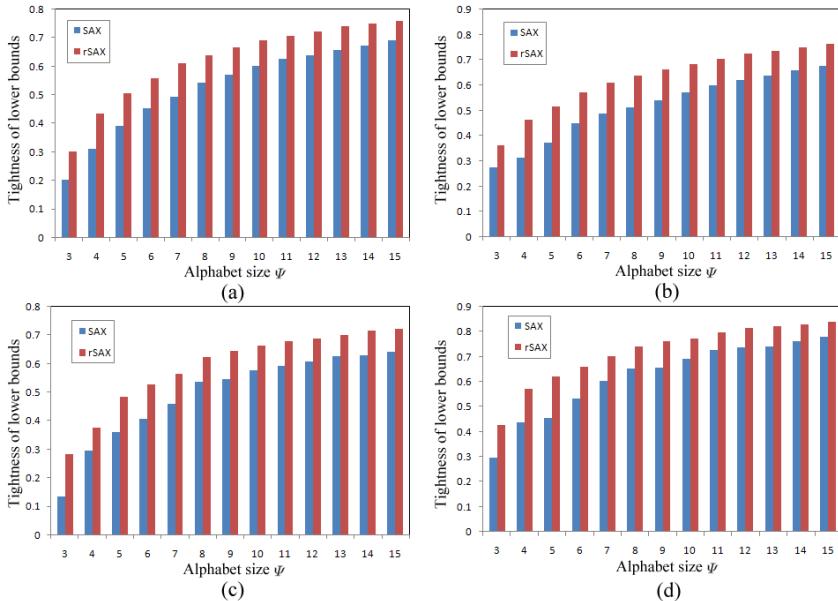


Fig. 5. Comparison of the tightness of lower bounds on 4 different time series data sets. (a) ECG 200 train data. (b) Gun Point Test data. (c) Koski ECG data. (d) Respiration data.

100 times, and calculated the average value of TLB for each alphabet size Ψ . For multiple time series, we just use the original full length as the length of Q and C , while for single time series, we sample subsequences with the length = 128. $\tau = 10$. Fig. 5 shows the results in 4 data sets, namely, the ECG 200 train data, the Gun Point Test data, the Koski ECG data, and the Respiration data. From this figure, we can find the TLB of rSAX outperforms that of SAX in all 4 data sets, especially when Ψ is small. For example, in Fig. 5 (c) $\Psi = 3$, the TLB of SAX is 0.1348, and the TLB of rSAX is 0.2834, which outperforms SAX about 210%.

5.2 Case Study by Finding Co-occurrent Abnormal Events

Here we carry out a case study of rSAX applications on a real-world data set from Meteorology. It is the historic data of daily temperatures series in Hunan Province, China [20]. The data set contains multiple temperature series from 97 different weather stations from 1998 to 2008. For each temperature series, there are 365×11 timestamps.

In meteorology, there are co-occurrent abnormal events happened in different places, thus cause some similar abnormal behaviors co-exist in multiple time series. Finding co-occurrent abnormal events can help experts to identify whether a co-occurrent unusual phenomenon is occurred by chance and thus implies whether this co-anomaly is of important for further analysis. The mining method including two steps: 1) using rSAX to represent multiple time series. After this step, similar points are more likely to be mapped to the same symbols; 2) applying statistical tests to find out all the subsequences

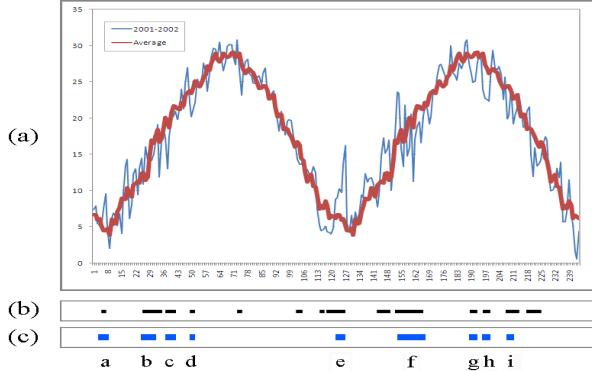


Fig. 6. (a) The average temperatures over two years, (b) the time periods of detected co-anomaly events, and (c) the time periods in which an event was recorded in the climate annual report

that are significantly abnormal in history. The ground truth comes from The Climate Reports of Hunan Province [21], which records the magnitude of influences and other descriptions of some abnormal climate events with big influences.

Fig. 6 shows the mining results from the year 2001-2002. Fig. 6 (a) shows the average temperature data of the year 2001-2002. Fig. 6 (b) shows the durations of detected co-occurred abnormal events, and (c) shows the recorded 9 abnormal events in Climate Reports. The x-axis is the time (one timestamp is the average of 3 days), and the y-axis is the temperature($^{\circ}\text{C}$). As it shown in Fig. 6, all of the 9 reported events are successfully detected, along with a few additional detections that were not written in the climate report. As we are detecting co-anomaly events that may only associated with sub-dimensions of multiple temperature series. Thus, some of the co-occurred abnormal events were not recorded in climate report because they had no wide influence in the province. The parameter settings for this result are, $\Psi = 8$; $\tau = 10$; $w = 122$; $\alpha = 0.01$ (the significant level); $\phi = 10$.

The number of extracted co-occurred abnormal patterns will be less if using SAX at the first step. Because some real co-occurred behaviors near the breakpoints are easily represented by different characters, thus are not significant in counts. Fig. 7 shows the number of significant subsequences(words) when increasing the sampling times τ . The SAX result is that when setting $\tau = 1$. Other parameter settings are $\Psi = 8$, $w = 122$. For rSAX that generates multiple versions of representations for subsequences, only one version with the lowest P-value is kept. From the Fig. 7 we can see that the number of detected significant subsequences increasing from 606 to 1048 when increasing τ from 1 to 10. It is because random shifting strategy can better differentiate dissimilar points while mapping similar points to the same symbols with high probabilities. When we further increasing τ from 10 to 50, the number of significant worlds increases slowly. Based on this experiment, $\tau = 10$ is recommended.

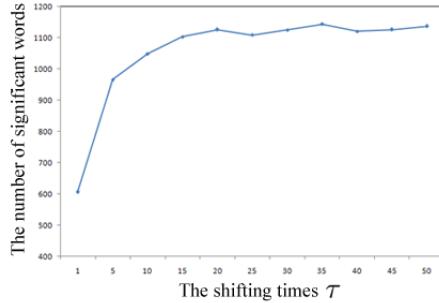


Fig. 7. The number of significant subsequences with respect to different shifting time

5.3 Parameter Settings

There are three parameters in rSAX, namely the alphabet size Ψ , the PAA length w and the shifting times τ . The first two parameters are the same with that of SAX, thus Ψ and w with bigger value will result finer-grained representations. The third parameter is the shifting times τ , in a special case when $\tau = 1$, there is no random shifting process in the mapping process, thus the rSAX method is equal to SAX. As shown in Fig. 8, we randomly selected two data sets for evaluation. It is easy to find that increasing the τ can improve the TLB.

Furthermore, we test the runtime of our algorithm with respect to different shifting times. Specifically, we run 30 times and take the average value for each parameter settings. Fig. 9 shows the runtime of rSAX when increasing τ on the Respiration data set (results on other data sets are similar). The PAA size $w = 128$, the alphabet size $\Psi = 6$. From this figure we can find that the runtime of rSAX is not sensitive to the parameter τ , especially when τ is less than 500.

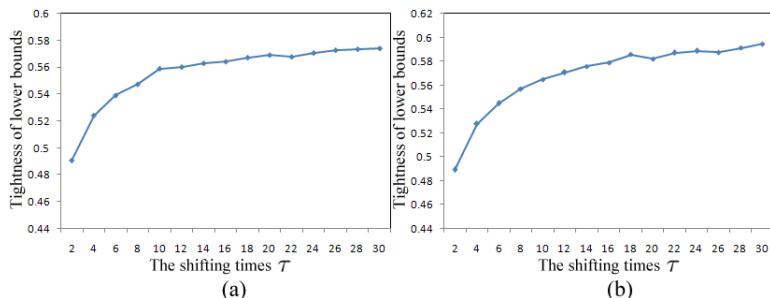


Fig. 8. The TLB results when increasing the sampling times. (a) On ECG 200 train data. (b) On Gun Point Test data.

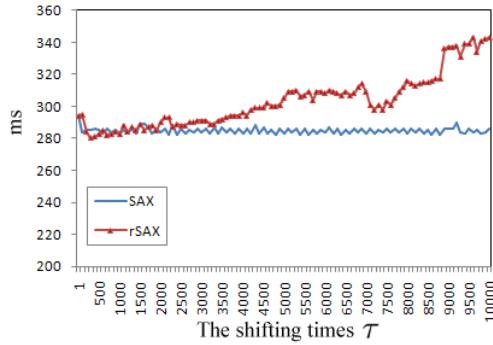


Fig. 9. The runtime comparisons of SAX and rSAX

6 Conclusion

In this paper, we proposed the rSAX approach for the symbolic representation of time series data. Compared with the traditional SAX approach, rSAX can significantly improve the tightness of lower bounds without increasing the alphabet size. Also, rSAX can guarantee the similar points have high probabilities to be represented by the same symbols. Finally, we theoretically and empirically validated the effectiveness and efficiency of the rSAX approach.

References

1. Lu, C.J., Lee, T.S., Chiu, C.C.: Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems* 47(2), 115–125
2. Yan, H., Pham, T.: Spectral similarity for analysis of dna microarray time-series data. *International Journal of Data Mining and Bioinformatics* 1(2), 150–161 (2006)
3. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter* 12(2), 74–82 (2011)
4. Han, J., Kamber, M.: *Data mining: concepts and techniques*. Morgan Kaufmann (2006)
5. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. In: *VLDB 2008* (2008)
6. Lin, J., Keogh, E., Patel, P., Lonardi, S.: Finding motifs in time series. In: *The 2nd Workshop on Temporal Data Mining* (July 2002)
7. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases, vol. 23. ACM (1994)
8. Chan, K.P., Fu, A.W.C.: Efficient time series matching by wavelets. In: *ICDE 1999*. IEEE (1999)
9. Keogh, E., Lin, J., Fu, A.: Hot sax: Efficiently finding the most unusual time series subsequence. In: *ICDM 2005* (2005)
10. Xi, X., Keogh, E., Wei, L., Mafra-Neto, A.: Finding motifs in a database of shapes. In: *SDM 2007* (2007)
11. Kasetty, S., Stafford, C., Walker, G.P., Wang, X., Keogh, E.: Real-time classification of streaming sensor data. In: *ICTAI 2008* (2008)
12. Camerra, A., Palpanas, T., Shieh, J., Keogh, E.: isax 2.0: Indexing and mining one billion time series. In: *ICDM 2010* (2010)

13. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems* 3(3), 263–286 (2001)
14. Huang, Y.W., Yu, P.S.: Adaptive query processing for time-series data. In: KDD 1999 (1999)
15. Megalooikonomou, V., Wang, Q., Li, G., Faloutsos, C.: A multiresolution symbolic representation of time series. In: ICDE 2005 (2005)
16. Wei, L., Keogh, E., Xi, X.: Saxually explicit images: Finding unusual shapes. In: ICDM 2006 (2006)
17. Shieh, J., Keogh, E.: isax: indexing and mining terabyte sized time series. In: KDD 2008 (2008)
18. Spencer, J.: The probabilistic method. In: Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 41–47 (1992)
19. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: The ucr time series classification/clustering homepage (2011)
20. Liao, Y., Wang, K., Zhao, F., Bai, S.: Modern agro-climatic zoning of Hunan Province. Hunan University Press, Changsha (2010)
21. The climate reports of hunan province, <http://www.hnqx.gov.cn>

Mining Frequent Itemsets from Sparse Data Streams in Limited Memory Environments

Juan J. Cameron¹, Alfredo Cuzzocrea², Fan Jiang¹, and Carson K. Leung¹

¹ University of Manitoba, Canada

{umcame33, umjian29, kleung}@cs.umanitoba.ca

² ICAR-CNR and University of Calabria, Italy

cuzzocrea@si.deis.unical.it

Abstract. Floods of data can be produced in many applications such as Web click streams or wireless sensor networks. Hence, algorithms for mining *frequent itemsets* from data streams are in demand. Many existing stream mining algorithms capture important streaming data and assume that the captured data can fit into main memory. However, problem arose when the available memory so limited that such an assumption does not hold. In this paper, we present a data structure called *DSTable* to capture important data from the streams onto the disk. The DSTable can be easily maintained and is applicable for mining frequent itemsets from streams (especially sparse data) in limited memory environments.

1 Introduction and Related Work

Over the past two decades, numerous frequent itemset (FI) mining algorithms have been proposed [5,6,11,15,17,18,20]. For example, FP-growth [10] uses an extended prefix-tree structure called *Frequent Pattern tree (FP-tree)* to capture the content of the database (DB). Although there are some works [2,9] that use disk-based structure for mining, they mostly mine FIs from *static* DBs. As a preview, we mine FIs from *dynamic* data streams.

The automation of measurements and data collection, together with the increasing development and use of a large number of sensors, have also led to streams of data [12,14,21]. To make sense of the streaming data, stream mining algorithms are needed. Mining from *dynamic* data streams (cf. static DBs) is more challenging due to the following *properties* of data streams:

Property 1: Data streams are continuous and unbounded. To find FIs from streams, we no longer have the luxury of performing multiple data scans. Once the streams flow through, we lose them. Hence, we need some data structures to capture the important contents of the streams (e.g., recent data—because users are usually more interested in recent data than older ones).

Property 2: Data in the streams are not necessarily uniformly distributed; their distributions are usually changing with time. A currently infrequent itemset may become frequent in the future, and vice versa. So, we have to be careful not to prune infrequent itemsets too early; otherwise, we may not be able to get

complete information such as frequencies of certain itemsets (as it is impossible to retract those pruned itemsets).

Algorithms for mining FIs from data streams have been proposed. For example, *approximate* algorithms (e.g., FP-streaming [8]) focus mostly on efficiency. Due to the approximate nature, these algorithms may find some infrequent itemsets or miss frequency information of some FIs (i.e., some false positives or negatives). An *exact* algorithm mines truly FIs (i.e., no false positives and no false negatives) by (i) constructing a *Data Stream Tree (DSTree)* [16] to capture global contents of the streaming data and then (ii) recursively building FP-trees locally for projected DBs based on the information extracted from the DSTree. Such an algorithm works well for situations where these global DSTree and local FP-trees can fit into main memory. When facing situations where not all local FP-trees can fit into memory, *Data Stream Projected trees (DSP-trees)* [13] can be built for x -projected DBs (where x is an item) as alternatives to local FP-trees built for α -projected DBs (where α is an itemset, i.e., $\alpha \supseteq x$).

Although memory is not too expensive nowadays, the amount of data generated in data streams also keeps growing at a much rapid rate. Algorithms for mining FIs with limited memory are still in demand. Preliminary results of our recent study [3] showed the feasibility of stream mining with limited memory. Our **key contribution** of the current paper is our simple yet powerful on-disk data structure called *DSTable* for capturing and maintaining relevant data found in the data streams. The DSTable is designed for *stream mining* of *FIs* using the window-sliding model [4,7,19]. When the streams flow through, a fixed-size window (i.e., a window containing the interesting portion of the streams—usually, recent data) slides and our DSTable is properly updated. Although we design the DSTable for environments with limited memory, it can be used as an *alternative structure* to the DSTree for environments with sufficient memory as well.

This paper is organized as follows. Next section presents our DSTable for capturing important information from data streams and describes how our DSTable can efficiently mine FIs from (sparse) data streams. Evaluation results and conclusions are shown in Sections 3 and 4, respectively.

2 Mining with the Gloabl DSTable

Recall from the previous section, FIs can be mined using (i) ⟨global DSTree, local FP-trees⟩ when these trees can fit into main memory and (ii) ⟨global DSTree, local DSP-trees⟩ where the memory is so limited that local FP-trees cannot fit. Here, we deal with situations where the memory is so limited that neither the global DSTree nor local DSP-trees can fit. Specifically, we propose a new (tabular) structure called **Data Stream Table (DSTable)** for mining data streams with limited memory environments.

Given (i) a stream of uncertain data and (ii) a limited memory environment, our proposed DS-Table structure captures the important contents of the streaming data onto the disk. Specifically, the DSTable is a two-dimensional table that captures the contents of transactions in all batches in the current sliding window.

Each row of the DSTable represents a domain item. Due to the dynamic nature and Property 2 of data streams, frequencies of items are continuously affected by the insertion of new batches (and removal of old batches) of streaming data. Arranging the items (i.e., rows of the DSTable) in frequency-dependent order may require frequent rearrangement. Hence, in the DSTable, items are arranged according to some canonical order (e.g., alphabetical order), which can be specified by the user prior to the construction of the DSTable. Consequently, the DSTable can be constructed using only a single scan of the data stream.

Each table entry is a pointer that points to the location of the table entry (e.g., itemID and column number) for the “next” item on the same transaction. As we are dealing with streaming data, we need to be able distinguish transactions in one batch from those in another batch. When the window slides, transactions in the old batch need to be removed and transactions in the new batch need to be appended.

Representation 1: Transaction IDs. A natural solution is to add a transaction ID to each table entry (i.e., $\langle TID, \text{itemID}, \text{column number} \rangle$). By keeping track of all transactions belonging to each batch, one can identify the transactions that are in the oldest batch when the window slides.

Example 1. Consider the following stream of transactions:

First batch (B_1)	$t_1=\{a\}$, $t_2=\{a, b, e\}$, $t_3=\{a, c, d\}$
Second batch (B_2)	$t_4=\{a, c, d, e\}$, $t_5=\{a, d\}$, $t_6=\{a, b\}$
Third batch (B_3)	$t_7=\{a, e\}$, $t_8=\{a, c, d, e\}$, $t_9=\{c\}$

Let user-specified $minsup$ threshold be 2 and the window size w be 2 batches (indicating that only two batches of transactions are kept). Then, we start constructing a DSTable such that each table entry is a triplet containing (i) TID t_i , (ii) itemID for the “next” item in the same t_i , and (iii) column number for the “next” item in the same t_i . Items in the DSTable are arranged in canonical order (e.g., e, d, c, b, a so as to facilitate fast upward traversal of DSP-trees). With this order, the transactions are scanned once and contents are put into the DSTable. The following shows that this representation DSTable (for sparse data), which captures the first two batches:

Row a: $(t_1, \emptyset, \emptyset)$, $(t_2, \emptyset, \emptyset)$, $(t_3, \emptyset, \emptyset)$, $(t_4, \emptyset, \emptyset)$, $(t_5, \emptyset, \emptyset)$, $(t_6, \emptyset, \emptyset)$
Row b: $(t_2, a, 2)$, $(t_6, a, 6)$
Row c: $(t_3, a, 3)$, $(t_4, a, 4)$
Row d: $(t_4, c, 1)$, $(t_4, c, 4)$, $(t_5, a, 5)$
Row e: $(t_2, b, 1)$, $(t_4, d, 2)$

Starting from the bottom (e.g., Row e) of the DSTable, we easily retrieve transaction $t_2 = \{a\}$ represented by $\langle (t_2, b, 1), (t_2, a, 2), (t_2, \emptyset, \emptyset) \rangle$. Here, the first component in these triplets indicates that they belong to t_2 . The second & third components tell the “next” item. For instance, $(t_2, b, 1)$ in Row e links e to Column 1 of b , which is linked to Column 2 of a . As a result, we get $\{e, b, a\}$. Afterwards (at time T'), when subsequent batches (e.g., the third batch B_3) of streaming data flow in, transactions t_1-t_3 are removed and t_7-t_9 are inserted into the DSTable. \square

Representation 2: Batch IDs. One potential concern observed from Example 1 is that one needs to keep extra mapping information between transaction IDs and batch IDs (e.g., t_1-t_3 belong to B_1). Hence, another representation is to use a batch ID. In other words, each table entry is of the form $\langle \text{batchID}, \text{itemID}, \text{column number} \rangle$. By so doing, we can easily identify all transactions belong to the oldest batch and we do not need to keep extra mapping information.

Example 2. We revisit Example 1. The following shows Representation 2 of the DSTable:

Row a: $(B_1, \emptyset, \emptyset), (B_1, \emptyset, \emptyset), \underline{(B_1, \emptyset, \emptyset)}, (B_2, \emptyset, \emptyset), (B_2, \emptyset, \emptyset), (B_2, \emptyset, \emptyset)$
Row b: $(B_1, a, 2), (B_2, a, 6)$
Row c: $(B_1, a, 3), (B_2, a, 4)$
Row d: $(B_2, c, 1), (B_2, c, 4), (B_2, a, 5)$
Row e: $(B_1, b, 1), (B_2, d, 2)$

Starting from the bottom (e.g., Row e) of the DSTable, we easily retrieve a transaction represented by $\langle (B_1, b, 1), (B_1, a, 2), (B_1, \emptyset, \emptyset) \rangle: \{e, b, a\}$. Then from Row c, we retrieve $\langle (B_1, a, 3), (B_1, \emptyset, \emptyset) \rangle$, which indicates $\{c, a\}$. Next, from Row a, we retrieve the unvisited triplet $\underline{(B_1, \emptyset, \emptyset)}$, which indicates $\{a\}$. When subsequent batches (e.g., the third batch B_3) of streaming data flow in, transactions in B_1 are removed and those in B_3 are inserted into the DSTable. \square

Representation 3: Pointers. On the positive side, Representation 2 saves us from keeping track of the mapping between batch ID and transaction ID. On the negative side, each table entry is still a triplet, which takes up lots of disk space. A better solution is not to store any transaction ID or batch ID. Instead, the DSTable stores pointers that point to the entries at the end of the batch (i.e., indicating the boundaries). When the window slides, one can easily remove every table entries between the previous (or the first) and this pointer.

Example 3. We revisit Example 1. The following shows Representation 3 of the DSTable:

Row a: Columns 3 and 6 $(\emptyset, \emptyset), (\emptyset, \emptyset), (\emptyset, \emptyset), (\emptyset, \emptyset), (\emptyset, \emptyset), (\emptyset, \emptyset)$
Row b: Columns 1 and 2 $(a, 2), (a, 6)$
Row c: Columns 1 and 2 $(a, 3), (a, 4)$
Row d: Columns 0 and 3 $(c, 1), (c, 4), (a, 5)$
Row e: Columns 1 and 2 $(b, 1), (d, 2)$

We observe some benefits with Representation 3. First, we can easily compute the frequency value for each domain items by counting the number of entries in the corresponding row (e.g., frequency of a is 4). As such, we can determine which domain items are frequent and which are not. Second, this representation is more compact when compared to the other two. In each table entry, only two components are kept. Third, the number of pointers are bounded by the number of domain items m and the window size w : The number of pointers is at most $O(m \times w)$. Fourth, when the window slides, we no longer need to read each row to remove transactions in older batches. We can jump to the beginning of the next batch. For instance, pointers in Row d (i.e., Columns 0 and 3) indicates that (i) no transaction involving d is in B_1 , (ii) $3 - 0 = 3$ transactions involving d are in B_2 , (iii) pairs representing new transactions involving d can be added starting from Column 4 = $3 + 1$.

Starting from the bottom (e.g., Row e) of the DSTable, we know from the pointer information that only $2 - 1 = 1$ transaction involving e in B_1 . We can easily retrieve a transaction $\{e, b, a\}$ in a similar fashion as in Example 2. Then from Row c, we retrieve $\{c, a\}$. Next, from Row a, we retrieve the unvisited pair (\emptyset, \emptyset) for $\{a\}$. \square

Mining with the Global DSTable and Local DSP-Trees. Since items are arranged in canonical order, we extract transactions (which we have shown in the above examples) during the mining process by locating the first frequent domain item y , extracting appropriate entries from Row y of the DSTable, and inserting these entries into a DSP-tree for y . Based on the user-defined $minsup$ threshold, we can determine which item is frequent. An item is frequent if its frequency or support value $\geq minsup$. So, during the path extraction and DSP-tree construction process, we just skip infrequent items. Afterwards, we repeat the same process for other frequent domain items.

3 Evaluation Results

Analytical Results. Mining with the DSTable uses a “delayed” mode (i.e., actual mining of FIs is delayed until it is needed to return to user the FIs). Hence, for $S=100$ batches, we only need to build a global DSTable and updates it 95 ($= S-w = 100-5$) times. Afterwards, we have an updated DSTable capturing the 96th to the 100th batches, and we build only $O(m)$ DSP-trees, where m is the number of frequent domain items in the domain, to find FIs—i.e., only one set of the updated a global FP-tree and m DSP-trees (cf. building 100 sets of a global tree and $O(f \times d)$ FP-trees by FP-streaming [8], one set for each of the 100 batches). Note that the number of frequent domain items is usually much less than the number of FIs: $m \ll f \leq 2^m - 1$. At any time during the mining process, only the global FP-tree and one DSP-tree are needed to be present (cf. one global FP-tree + $O(d)$ subsequent FP-trees are needed to be present in FP-streaming).

Moreover, as the DSTable can reside on disk (thus, serving as an alternative to FP-tree when memory is limited), the size of the DSTable is independent of minsup . Hence, it is useful for interactive mining, especially when users keep adjusting minsup . Please note that the DSTable captures the transactions in the current sliding window. During the mining process, the algorithm skips infrequent items (i.e., items having support lower than minsup) and only includes frequent items when building DSP-trees. When users adjust minsup during the interactive mining process, we do not need to rebuild the DSTable.

Experimental Results. We used different datasets such as IBM synthetic data, which are generated by the program developed at IBM Almaden Research Centre [1]. The data contain 1M records with an average transaction length of 10 items, and a domain of 1,000 items. We set each batch to be 0.1M transactions and the window size $w=5$ batches. All experiments were run in a time-sharing environment in a 1 GHz machine. The reported figures are based on the average of multiple runs. Runtime includes CPU and I/Os; it includes the time for both

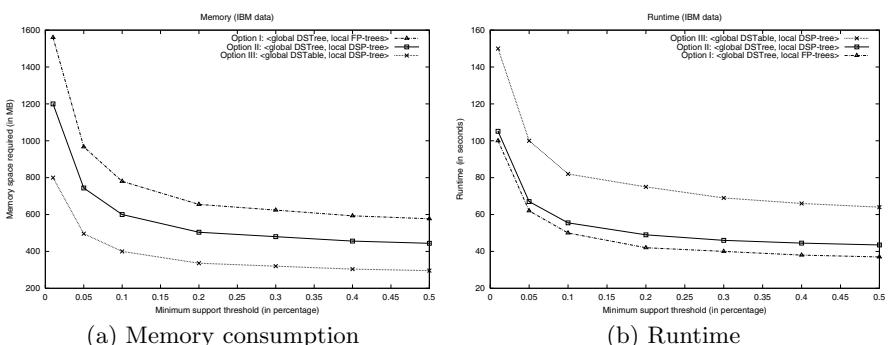


Fig. 1. Experimental results of our proposed DSTable for IBM data (sparse data)

tree construction and FI mining steps. In the experiments, we mainly evaluated the accuracy and efficiency of the DSTable.

First, we measured the accuracy of the three mining options: (i) \langle global DSTree, local FP-trees \rangle , (ii) \langle global DSTree, local DSP-trees \rangle , and (iii) \langle global DSTable, local DSP-trees \rangle options. Experimental results show that mining with any of these three options give the same mining results.

Then, we measured the space and time efficiency of our proposed DSTable. Fig. 1(a) shows that the \langle DSTable, DSP-trees \rangle option required the *smallest* main memory space because our proposed DSTable is a disk-based structure. Fig. 1(b) shows the \langle DSTable, DSP-trees \rangle option took slightly longer because it needs to read from disk whereas the former two just read from main memory. However, reading from disk would be a *logical choice* in a limited main memory environment.

Then, we tested with the usual experiment (e.g., the effect of *minsup*). As shown in Fig. 1(b), the runtime decreased when *minsup* increased. Moreover, mining with our proposed DSTable was scalable with respect to the number of transactions.

4 Conclusions

A key contribution of this paper is to provide the user with a simple yet powerful alternative structure for efficient FI mining from sparse data streams in limited memory environments using sliding windows. Our DSTable captures the contents of transactions in a window, and arranges items according to some canonical order that is unaffected by changes in item frequency when the window slides. To avoid the recursive construction of trees during the mining process, DSP-trees are built from the DSTable.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB 1994, pp. 487–499 (1994)
2. Buehrer, G., Parthasarathy, S., Ghoting, A.: Out-of-core frequent pattern mining on a commodity. In: ACM KDD 2006, pp. 86–95 (2006)
3. Cameron, J.J., Cuzzocrea, A., Leung, C.K.-S.: Stream mining of frequent sets with limited memory. In: ACM SAC 2013, pp. 173–175 (2013)
4. Cao, K., Wang, G., Han, D., Ma, Y., Ma, X.: A framework for high-quality clustering uncertain data stream over sliding windows. In: Gao, H., Lim, L., Wang, W., Li, C., Chen, L. (eds.) WAIM 2012. LNCS, vol. 7418, pp. 308–313. Springer, Heidelberg (2012)
5. Chiu, D.Y., Wu, Y.H., Chen, A.: Efficient frequent sequence mining by a dynamic strategy switching algorithm. VLDB J. 18(1), 303–327 (2009)
6. Fariha, A., Ahmed, C.F., Leung, C.K.-S., Abdullah, S.M., Cao, L.: Mining frequent patterns from human interactions in meetings using directed acyclic graphs. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013, Part I. LNCS (LNAI), vol. 7818, pp. 38–49. Springer, Heidelberg (2013)

7. Gao, C., Wang, J., Yang, Q.: Efficient mining of closed sequential patterns on stream sliding window. In: IEEE ICDM 2011, pp. 1044–1049 (2011)
8. Giannella, C., Han, J., Pei, J., Yan, X., Yu, P.S.: Mining frequent patterns in data streams at multiple time granularities. In: Data Mining: Next Generation Challenges and Future Directions, ch. 6 (2004)
9. Grahne, G., Zhu, J.: Mining frequent itemsets from secondary memory. In: IEEE ICDM 2004, pp. 91–98 (2004)
10. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD 2000, pp. 1–12 (2000)
11. Jiang, X., Xiong, H., Wang, C., Tan, A.-H.: Mining globally distributed frequent subgraphs in a single labeled graph. DKE 68(10), 1034–1058 (2009)
12. Jin, R., Agrawal, G.: An algorithm for in-core frequent itemset mining on streaming data. In: IEEE ICDM 2005, pp. 210–217 (2005)
13. Leung, C.K.-S., Brajczuk, D.A.: Efficient mining of frequent itemsets from data streams. In: Gray, A., Jeffery, K., Shao, J. (eds.) BNCOD 2008. LNCS, vol. 5071, pp. 2–14. Springer, Heidelberg (2008)
14. Leung, C.K.-S., Cuzzocrea, A., Jiang, F.: Discovering frequent patterns from uncertain data streams with time-fading and landmark models. In: Hameurlain, A., Küng, J., Wagner, R., Cuzzocrea, A., Dayal, U. (eds.) TLDKS VIII. LNCS, vol. 7790, pp. 174–196. Springer, Heidelberg (2013)
15. Leung, C.K.-S., Hayduk, Y.: Mining frequent patterns from uncertain data with mapReduce for big data analytics. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013, Part I. LNCS, vol. 7825, pp. 440–455. Springer, Heidelberg (2013)
16. Leung, C.K.-S., Khan, Q.I.: DSTree: a tree structure for the mining of frequent sets from data streams. In: IEEE ICDM 2006, pp. 928–932 (2006)
17. Leung, C.K.-S., Tanbeer, S.K.: PUF-tree: a compact tree structure for frequent pattern mining of uncertain data. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013, Part I. LNCS (LNAI), vol. 7818, pp. 13–25. Springer, Heidelberg (2013)
18. Qu, J.-F., Liu, M.: A high-performance algorithm for frequent itemset mining. In: Gao, H., Lim, L., Wang, W., Li, C., Chen, L. (eds.) WAIM 2012. LNCS, vol. 7418, pp. 71–82. Springer, Heidelberg (2012)
19. Papapetrou, O., Garofalakis, M., Deligiannakis, A.: Sketch-based querying of distributed sliding-window data streams. In: VLDB 2012, pp. 992–1003 (2012)
20. Tanbeer, S.K., Leung, C.K.-S.: Finding diverse friends in social networks. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 301–309. Springer, Heidelberg (2013)
21. Tirthapura, S., Woodruff, D.P.: A general method for estimating correlated aggregates over a data stream. In: IEEE ICDE 2012, pp. 162–173 (2012)

Human Dynamics Revealed through Log Analytics in a Cloud Computing Environment

Sixi Chen¹, Ning An^{1,*}, Lian Li¹, Yongwei Wu², Weimin Zheng², and Lin Sun³

¹ Gerontechnology Lab, Hefei University of Technology, Hefei, China

chensixi3006@sina.com, ning.g.an@acm.org, llian@hfut.edu.cn
² Tsinghua National Laboratory for Information Science and Technology, Tsinghua University,
Beijing, China

{wuyw, zwm-dcs}@tsinghua.edu.cn

³ Gansu Institute of Political Science and Law, Gansu, China
s16684@gsli.edu.cn

Abstract. Researchers have used the Web to better observe, monitor and reason various aspects of human behaviors. In the growing popularity of Cloud Computing environments, however, this important topic has yet to be carefully investigated. In this paper, we analyze 5-month logs detailing the access history to MeePo system that is a private cloud storage service deployed at Hefei University of Technology providing storage services to its college students. We find interesting activity patterns both for individual users and entire population. In particular, we find that the real world phenomena can be reflected closely in MeePo system.

Keywords: human dynamics, log analytics, cloud computing.

1 Introduction

Human behaviors have drawn great interests from researchers in various academic disciplines, including psychology, sociology, economics and anthropology. Gaining insights into human behaviors, however, remains a challenge because of the underlying complexity and intangibility. Given web technologies have permeated into nearly every aspect of society, people have started to model and understand human behaviors through readily available massive data on the web. For instance, researchers have worked on better understanding human navigation patterns in terms of visiting web pages and communication patterns between users and web sites so that they could enhance user experiences of common navigational activities [1][2]. Furthermore, popular online social networks, including Facebook, Twitter and Sina Weibo, provide yet another kind of platforms for observing human behaviors. Gaining extra insights into human behaviors through describing and analyzing human online behaviors mathematically could have important implications in online resource management, service allocation and other applications.

* Corresponding author.

One frequently observed form of human dynamics is temporal patterns generated by human behaviors. Conventionally, people assumed that the time property of an individual's activity pattern following a Poisson process. Recent theoretical analyses and empirical studies, however, showed that the timing of various human activities follows the power law at both the entire population level and an individual level [3][4][5][6][7]. In particular, according to Barabási [3], many human activities have burst nature that can be modeled as a task-driven queuing process with a highest-priority-first (HPF) protocol deciding which task needs to be executed first. On the other hand, researchers have looked into using interest-based models [4], which also show heavy-tailed distribution, to explain human activity patterns: watching online movies [5], browsing the web [6], and playing online games [7].

Even in the latest computing paradigm, Cloud Computing, people started to investigate various ways to utilize human behaviors. Viewing computation as a utility, Cloud Computing, has garnered a great interest from people from different domains because of its unique features including elasticity, on demand service and pay-as-you-go. The Cloud Computing system, however, is prone to performance anomalies due to its inherent complexity. To overcome this obstacle, people have conducted user behavior analyses at different levels of Cloud Computing system, including cloud computing system performance, security, and deployment strategies [8][9]. Amazon cloud services also benefitted from improving the custom relationship by tracking and monitoring activities of each individual user. Few people, however, have looked into the relationship between human dynamics in the real world and their behaviors in Cloud Computing environment. In this paper, we aim to investigate this topic by studying human dynamics of college students at Hefei University of Technology through log analytics in a private storage cloud called MeePo.

2 Cloud Computing Platform: MeePo System

MeePo system, providing private cloud storage service, is designed and implemented by Tsinghua National Laboratory for Information Science and Technology of Tsinghua University. MeePo is targeted at staff and students in a university providing stable storage service freely. In particular, MeePo system offers three types of services: public storage service, community storage service and individual storage service [11]. More specifically, public space mainly provides data sharing service for all MeePo users; community space offers data storage service and data sharing service for a specific group of users; private space just serves an individual user with certain privileges. MeePo system has been successfully running in Tsinghua University, Lanzhou University and Hefei University of Technology.

Our experimental data are server log files collected come from MeePo system deployed at Hefei University of Technology which we name it HFUT-MeePo for convenience. Overall, HFUT-MeePo offers three types of storage services: 1) public space, including 7 categories: study, movie, TV, music, comic, game and software; 2) community space, registered and accessed by a group of users, who share similar interests or belong to the same association; and 3) private space, storing personal data.

Experimental data were collected over a 159-day period that goes from 24th May, 2012 to 31st October, 2012, in which two-day data are excluded because of power failure. The log file format of HFUT-MeePo is: [time; function responses to user operations; user operation; user email; requested resource location].

3 Activity Pattern of the Population

In this section, we study the activity pattern of the entire user population of HFUT-MeePo system. At this time, there are 3,000 users that have registered and utilized MeePo cloud service. All log files stored in cloud platform are collected in order to analyze what user behaviors in terms of daily, weekly and monthly activities. How user behaviors may correspond to regular school affairs including statutory holidays for university staff and students are also studied.

Taking number of service accesses as a measurement unit, we look into how this measurement changes during a week. In general, with the unceasing change of the busy degree in a week, the number of service access will change correspondingly; and people often associate Monday as the busiest and worst day of a week. However, it doesn't seem to be the case. A survey suggests that it's worse off on Tuesday than Monday, both in terms of stress levels and workload [12]. In our analysis, the weekly average of the number of service access during the whole period is shown in Fig. 1. It shows that Tuesday is the least active day in a week, while the activity degree is increasing from then on and reaches a peak on Thursday. Interestingly, the level of activity on Friday and Saturday is similar to that on Wednesday. Meanwhile, we also provide the variation trend of the number of service access during the holiday break of Mid-Autumn Festival and National Day. As shown in Fig. 1, the first green dotted line represents the Mid-Autumn Day, followed by the National Day, a seven-day-holiday. It's worthwhile to note that the trend of this period is different when compared with the weekly average before the Mid-Autumn Day. Furthermore, we observe that there was a sudden decrease in the first two days of the National Day holidays and the quietest period is actually in the middle of the holidays. This maps to the physical world pretty closely: most users started to leave campus when the National Days started and gradually returned to campus and started to use HFUT-MeePo system again towards the end of the holidays.

Human circadian rhythms, a human nature, are one of reasons for inhomogeneity of human behaviors [10]. In order to study human circadian rhythms in HFUT-MeePo system, we look into how user behaviors change from one hour to another in a day by averaging the user visits hourly over all weeks as shown in Fig. 2. We can see that on workdays, the 12th Hour and the 13th Hour are the most active period due to the lunch break and with the activity gradually increasing from the 16th Hour until midnight. As a whole, the trend shown in Fig. 2 well reflects the circadian rhythms of the college students at HFUT. More students stay late on Saturdays between the 0th Hour to the 7th Hour than those on Workdays and Sundays. Correspondingly, Saturdays see fewer students active between the 14th Hour to the 19th Hour compared to those on Workdays and Sundays. On Sundays, MeePo system sees more user activities because people are getting ready for the coming week.

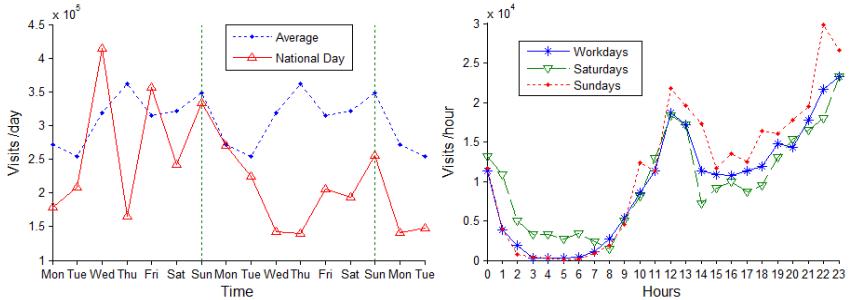


Fig. 1. A Comparison of average weekly activity and the activity during the break of Mid-Autumn Day and National Day

Fig. 2. Average hourly activity on workdays, Saturdays and Sundays

4 Individual User Skew the Behavior

In the preceding section, we discuss the collective behavior of the population during 159 days. To shed more lights on activity patterns, we inspect individual activities in this section. When we observe seven categories of Public space in HFUT-MeePo system, there are several significant activity peaks in the total visits during the whole study period, specifically for the Study category which is one of the seven categories of Public space. One peak occurred between June 15th and July 1st. Intuitively, we thought that some important things about learning might occur at that time. However, through careful analyses of the 14 areas of Study category, we find that the peak is caused by one individual. On the basis of the amount of Read operation that this user did, we ascertain that the cause of the high-traffic is not evoked by some kinds of programs and it is further confirmed by the communication with this user via email.

To further investigate the skew phenomenon of individual activities, we utilize other statistical analysis methods. In particular, we represent the histograms of the time interval between two consecutive visits of a single user to MeePo in Fig. 3. Fig. 3(a) shows the probability dense function (PDF) of time interval between two consecutive visits of one user. Meanwhile, Fig. 3(b) shows the cumulative distribution function $P(t)$ and the power-law fit $P_r(t)$. The cumulative distribution of session length can be well fitted by a power-law, $P(t) \sim t^{-2.21}$ (the dotted line in the log-log pot has the slope -2.21), which means the distribution of session length fits the power-law with the exponent -1.21. Note that the total number of events is 50 during 10^5 time steps.

The burst nature of human dynamics was first proposed in 2005 by Albert-László Barabási [3]. A lot of evidences indicate that the timing of numerous human activities follows non-Poisson statistics, and is characterized by bursts of rapidly occurring events separated by a long time of inactivity. Using the queuing process, Barabási pointed out that the timing of the tasks with perceived priority is heavy tailed: most tasks will be executed rapidly, while few tasks with low priority will wait a long time. In addition to the task-based mechanism, the interest-based model is proposed in [13].

In this model, the quasi-periodic change of interest will affect the frequency of events, and vice versa. It can generate the power-law inter-event time distribution with exponent of value -1 based on the interest. In our study, Fig. 3 (b) can be well illustrated according to the interest-based model. When a user browses data stored in MeePo system, we prefer to think that this user's behavior is based on his/her personal preferences. If some files of a particular group are attractive to a user, this user is likely to visit other documents in this group; on the other hand, this user's interests, i.e. visits, will continue grow until reaching a saturation point. Once this happens, visits will be reduced significantly for a prolonged period of time. Hence, analysis to the time interval between consecutive visits can contribute to a better understanding of human dynamic.

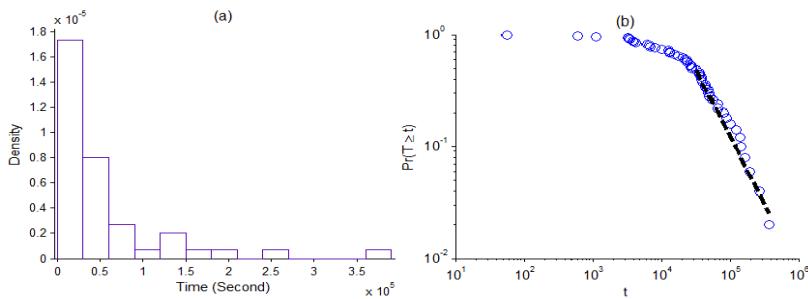


Fig. 3. Distribution of time interval between two consecutive visits of one user: (a) the probability dense function, (b) cumulative distribution function $P(t)$ and power-law fit $Pr(t)$

5 Discussion and Conclusions

In this paper, we study human dynamics by analyzing log files of MeePo system, a private cloud storage service, deployed at Hefei University of Technology. First of all, we investigate the activity pattern of the entire population. We compare the MeePo service access trend between the weekly average and the holiday week during the break of Mid-Autumn festival and National Day holidays. Analysis results indicate that how people spend their holidays in the physical world reflected in their online activities which show rather different trend comparing to the regular week. Furthermore, analyzing the weekly average activities and hourly average activities clearly reveal human circadian rhythms and weekly cycles. Second, because of our own observation of the burst of visited caused by one user, we investigate the interval time between consecutive events based on Barabási's theory. We find that visits to MeePo system follow interest-based model instead of Barabási's task-driven model.

In the near future, we plan to collect and analyze more data from MeePo system deployed at other universities, including Tsinghua University and Lanzhou University. We can then explore whether human dynamics revealed through log analytics vary from one campus to another.

Acknowledgements. This work was supported in part by the National High Technology Research and Development Program of China (863 Program) under grant 2012AA011005, by Chinese Special Project of Science and Technology (2012ZX01039001), by the Chinese National Science and Technology Program for Public Wellbeing under Grant No. 2012GS620302, and by Hefei University of Technology under Grant 407-037036, 2011HGZY0018, and 2012HGZY0031.

References

1. Gonçalves, B., Ramasco, J.J.: Human dynamics revealed through Web analytics. *Phys. Rev. E* 78(2), 026123 (2008)
2. Cockburn, A., Mckenzie, B.: What do web users do? An empirical analysis of web use. *International Journal of Human-Computer Studies* 54, 903–922 (2001)
3. Barabási, A.L.: The origin of bursts and heavy tails in human dynamics. *Nature* 435, 207–211 (2005)
4. Han, X., Zhou, T., Wang, B.: Modeling human dynamics with adaptive interest. *New Journal of Physics* 10, 073010 (2008)
5. Zhou, T., Kiet, H., Kim, B.J., Wang, B., Holme, P.: Role of activity in human dynamics. *The European Physical Journal* 82, 28002 (2008)
6. Dezsö, Z., Almaas, Z., Lukács, A., Rácz, B., Szakadát, I., Barabási, A.L.: Dynamics of information access on the web. *Physical Review E* 73, 066132 (2006)
7. Henderson, T., Bhatti, S.: Modelling user behaviour in networked games. In: *Proceedings of the 9th ACM Multimedia Conference*, pp. 212–220. ACM Press, Canada (2001)
8. Montes, J., Nicolae, B., Antoniu, G., Sanchez, A., Perez, M.S.: Using Global Behavior Modeling to Improve QoS in Cloud Data Storage Services. In: *CLOUDCOM 2010 Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pp. 304–311 (2010)
9. Dean, D.J., Nguyen, H., Gu, X.: UBL: Unsupervised Behavior Learning for Prediction Performance Anomalies in Virtualized Cloud Systems. In: *ICAC 2012 Proceedings of the 9th International Conference on Autonomic Computing*, pp. 191–200 (2012)
10. Jo, H.H., Karsai, M., Kertész, J., Kaski, K.: Circadian pattern and burstiness in mobile phone communication. *New Journal of Physics* 14, 013055 (2012)
11. Wang, Q., Chen, K., Wu, Y., Zheng, W.: Improving the Effective IO Throughput by Adaptive Read-Ahead Strategy for Private Cloud Storage Service. In: *2012 Seventh ChinaGrid Annual Conference (ChinaGrid)*, Beijing, pp. 134–141 (2012)
12. Tuesday at 11:45 is most stressful time of the week, survey suggests, <http://www.telegraph.co.uk/news/newstopics/howaboutthat/5113653/Tuesday-at-1145-is-most-stressful-time-of-the-week-survey-suggests.html>
13. Guo, J., Fan, C., Guo, Z.: Weblog patterns and human dynamics with decreasing interest. *The European Physical Journal B* 81, 341–344 (2011)

Data Fusion: Resolving Conflicts from Multiple Sources

Xin Luna Dong¹, Laure Berti-Equille², and Divesh Srivastava³

¹ Google Inc.

lunadong@google.com

² Institut de Recherche pour le Developpement (IRD)

Laure.Berti@ird.fr

³ AT&T Labs-Research

divesh@research.att.com

Abstract. Many data management applications, such as setting up Web portals, managing enterprise data, managing community data, and sharing scientific data, require integrating data from multiple sources. Each of these sources provides a set of values and different sources can often provide conflicting values. To present quality data to users, it is critical to resolve conflicts and discover values that reflect the real world; this task is called *data fusion*. This paper describes a novel approach that finds true values from conflicting information when there are a large number of sources, among which some may copy from others. We present a case study on real-world data showing that the described algorithm can significantly improve accuracy of truth discovery and is scalable when there are a large number of data sources.

1 Introduction

The amount of useful information available on the Web has been growing at a dramatic pace in recent years. In a variety of domains, such as science, business, technology, arts, entertainment, politics, government, sports, tourism, there are a huge number of data sources that seek to provide information to a wide spectrum of information users. In addition to enabling the availability of useful information, the Web has also eased the ability to publish and spread false information across multiple sources. Widespread availability of conflicting information (some true, some false) makes it hard to separate the wheat from the chaff. Simply using the information that is asserted by the largest number of data sources (*i.e.*, naive voting) is clearly inadequate since biased (and even malicious) sources abound, and plagiarism (*i.e.*, copying without proper attribution) between sources may be widespread. *Data fusion* aims at resolving conflicts from different sources and find values that reflect the real world.

Ideally, when applying voting, we would like to give a higher vote to more trustworthy sources and ignore copied information; however, this raises many challenges. First, we often do not know *a priori* the trustworthiness of a source and that depends on how much of its provided data are correct, but the correctness of data, on the other hand, needs to be decided by considering the number and trustworthiness of the providers; thus, it is a chicken-and-egg problem. Second, in many applications we do not know how each source obtains its data, so we have to discover copiers from a snapshot of data.

Table 1. The motivating example: five data sources provide information on the affiliations of five researchers. Only S_1 provides all true values.

	S_1	S_2	S_3	S_4	S_5
<i>Stonebraker</i>	MIT	Berkeley	MIT	MIT	MS
<i>Dewitt</i>	MSR	MSR	UWisc	UWisc	UWisc
<i>Bernstein</i>	MSR	MSR	MSR	MSR	MSR
<i>Carey</i>	UCI	AT&T	BEA	BEA	BEA
<i>Halevy</i>	Google	Google	UW	UW	UW

The discovery is non-trivial: sharing common data does not in itself imply copying—accurate sources can also share a lot of independently provided correct data; not sharing a lot of common data does not in itself imply no-copying—a copier may copy only a small fraction of data from the original source; even when we decide that two sources are dependent, it is not always obvious which one is a copier. Third, a copier can also provide some data by itself or verify the correctness of some of the copied data, so it is inappropriate to ignore all data it provides.

In this paper, we present novel approaches for data fusion. First, we consider *copying* between data sources in truth discovery. Our technique considers not only whether two sources share the same values, but also whether the shared values are true or false. Intuitively, for a particular object, there are often multiple distinct false values but usually only one true value. Sharing the same true value does not necessarily imply copying between sources; however, sharing the same false value is typically a low-probability event when the sources are fully independent. Thus, if two data sources share a lot of false values, copying is more likely. Based on this analysis, we describe Bayesian models that compute the probability of copying between pairs of data sources and take the result into consideration in truth discovery.

Second, we also consider *accuracy* in voting: we trust an accurate data source more and give values that it provides a higher weight. This method requires identifying not only if two sources are dependent, but also which source is the copier. Indeed, accuracy in itself is a clue of direction of copying: given two data sources, if the accuracy of their common data is highly different from that of one of the sources, that source is more likely to be a copier.

Example 1. Consider the five data sources in Table 1. They provide information on affiliations of five researchers and only S_1 provides all correct data. Sources S_4 and S_5 copy their data from S_3 , and S_5 introduces certain errors during copying.

First consider the three sources S_1, S_2 , and S_3 . For all researchers except *Carey*, a naive voting on data provided by these three sources can find the correct affiliations. For *Carey*, these sources provide three different affiliations, resulting in a tie. However, if we take into account that the data provided by S_1 is more accurate (among the rest of the 4 researchers, S_1 provides all correct affiliations, whereas S_2 provides 3 and S_3 provides only 2 correct affiliations), we will consider *UCI* as most likely to be the correct value.

Now consider in addition sources S_4 and S_5 . Since the affiliations provided by S_3 are copied by S_4 and S_5 , naive voting would consider them as the majority and so make wrong decisions for three researchers. Only if we ignore the values provided by S_4 and S_5 , we will be able to again decide the correct affiliations. Note however that identifying the copying relationships is not easy: while S_3 shares 5 values with S_4 and 4 values with S_5 , S_1 and S_2 also share 3 values, more than half of all values. If we knew which values are true and which are false, we would suspect copying between S_3 , S_4 and S_5 , because they provide the same false values. On the other hand, we would suspect the copying between S_1 and S_2 much less, as they share only true values.

The structure of the rest of the paper is as follows. Section 2 presents how we can leverage source accuracy in data fusion. Section 3 presents how we can leverage copying relationships in data fusion. Section 4 presents a case study of these techniques on a real-world data set, and Section 5 concludes.

2 Fusing Sources Considering Accuracy

We first formally describe the data fusion problem and describe how we leverage the trustworthiness of sources in truth discovery. In this section we assume no-copying between data sources and defer discussion on copying to the next section.

2.1 Data Fusion

We consider a set of *data sources* \mathcal{S} and a set of *objects* \mathcal{O} . An object represents a particular aspect of a real-world entity, such as the affiliation of a researcher; in a relational database, an object corresponds to a cell in a table. For each object $O \in \mathcal{O}$, a source $S \in \mathcal{S}$ can (but not necessarily) provide a *value*. Among different values provided for an object, one correctly describes the real world and is *true*, and the rest are *false*. In this paper we solve the following problem: given a snapshot of data sources in \mathcal{S} , decide the true value for each object $O \in \mathcal{O}$.

We note that a value provided by a data source can either be atomic, or a set or list of atomic values (*e.g.*, author list of a book). In the latter case, we consider the value as true if the atomic values are correct and the set or list is complete (and order preserved for a list). This setting already fits many real-world applications and we refer our readers to [13] for solutions that treat a set or list of values as multiple values.

We consider a core case that satisfies the following two conditions (relaxation of these assumptions is discussed in [7]):

- *Uniform false-value distribution*: For each object, there are multiple false values in the underlying domain and an independent source has the same probability of providing each of them.
- *Categorical value*: For each object, values that do not match exactly are considered as completely different.

Note that this problem definition focuses on *static* information that does not evolve over time, such as authors and publishers of books, and we refer our readers to [8] for data fusion for evolving values.

2.2 Accuracy of a Source

Let $S \in \mathcal{S}$ be a data source. The *accuracy* of S , denoted by $A(S)$, is the fraction of true values provided by S ; it can also be considered as the probability that a value provided by S is the true value.

Ideally we should compute the accuracy of a source as it is defined; however, in real applications we often do not know for sure which values are true, especially among values that are provided by similar number of sources. Thus, we compute the accuracy of a source as the average probability of its values being true (we describe how we compute such probabilities shortly). Formally, let $\bar{V}(S)$ be the values provided by S and denote by $|\bar{V}(S)|$ the size of $\bar{V}(S)$. For each $v \in \bar{V}(S)$, we denote by $P(v)$ the probability that v is true. We compute $A(S)$ as follows.

$$A(S) = \frac{\sum_{v \in \bar{V}(S)} P(v)}{|\bar{V}(S)|}. \quad (1)$$

We distinguish *good* sources from *bad* ones: a data source is considered to be good if for each object it is more likely to provide the true value than any *particular* false value; otherwise, it is considered to be bad. Assume for each object in \mathcal{O} the number of false values in the domain is n . Then, in the core case, the probability that S provides a true value is $A(S)$ and that it provides a particular false value is $\frac{1-A(S)}{n}$. So S is good if $A(S) > \frac{1-A(S)}{n}$ (*i.e.*, $A(S) > \frac{1}{1+n}$). We focus on good sources in the rest of this paper, unless otherwise specified.

2.3 Probability of a Value Being True

Now we need a way to compute the probability that a value is true. Intuitively, the computation should consider both how many sources provide the value and accuracy of those sources. We apply a Bayesian analysis for this purpose.

Consider an object $O \in \mathcal{O}$. Let $\mathcal{V}(O)$ be the domain of O , including one true value and n false values. Let \bar{S}_o be the sources that provide information on O . For each $v \in \mathcal{V}(O)$, we denote by $\bar{S}_o(v) \subseteq \bar{S}_o$ the set of sources that vote for v ($\bar{S}_o(v)$ can be empty). We denote by $\Psi(O)$ the observation of which value each $S \in \bar{S}_o$ votes for O .

To compute $P(v)$ for $v \in \mathcal{V}(O)$, we need to first compute the probability of $\Psi(O)$ conditioned on v being true. This probability should be that of sources in $\bar{S}_o(v)$ each providing the true value and other sources each providing a particular false value:

$$\begin{aligned} Pr(\Psi(O) | v \text{ true}) &= \prod_{S \in \bar{S}_o(v)} A(S) \cdot \prod_{S \in \bar{S}_o \setminus \bar{S}_o(v)} \frac{1 - A(S)}{n} \\ &= \prod_{S \in \bar{S}_o(v)} \frac{nA(S)}{1 - A(S)} \cdot \prod_{S \in \bar{S}_o} \frac{1 - A(S)}{n}. \end{aligned} \quad (2)$$

Among the values in $\mathcal{V}(O)$, there is one and only one true value. Assume our *a priori* belief of each value being true is the same, denoted by β . We then have

$$Pr(\Psi(O)) = \sum_{v \in \mathcal{V}(O)} \left(\beta \cdot \prod_{S \in \bar{S}_o(v)} \frac{nA(S)}{1 - A(S)} \cdot \prod_{S \in \bar{S}_o} \frac{1 - A(S)}{n} \right). \quad (3)$$

Applying the Bayes Rule leads us to

$$P(v) = \Pr(v \text{ true} | \Psi(O)) = \frac{\prod_{S \in \bar{S}_o(v)} \frac{nA(S)}{1-A(S)}}{\sum_{v_0 \in \mathcal{V}(O)} \prod_{S \in \bar{S}_o(v_0)} \frac{nA(S)}{1-A(S)}}. \quad (4)$$

To simplify the computation, we define the *confidence* of v , denoted by $C(v)$, as $C(v) = \sum_{S \in \bar{S}_o(v)} \log \frac{nA(S)}{1-A(S)}$. If we define the *accuracy score* of a data source S as $A'(S) = \log \frac{nA(S)}{1-A(S)}$, we have $C(v) = \sum_{S \in \bar{S}_o(v)} A'(S)$. So we can compute the confidence of a value by summing up the accuracy scores of its providers. Finally, we can compute the probability of each value as $P(v) = \frac{2^{C(v)}}{\sum_{v_0 \in \mathcal{V}(O)} 2^{C(v_0)}}$. A value with a higher confidence has a higher probability to be true; thus, rather than comparing vote counts, we can just compare confidence of values. The following theorem shows three nice properties of Equation (4).

Theorem 1. *Equation (4) has the following properties:*

1. *If all data sources are good and have the same accuracy, when the size of $\bar{S}_o(v)$ increases, $C(v)$ increases;*
2. *Fixing all sources in $\bar{S}_o(v)$ except S , when $A(S)$ increases for S , $C(v)$ increases.*
3. *If there exists $S \in \bar{S}_o(v)$ such that $A(S) = 1$ and no $S' \in \bar{S}_o(v)$ such that $A(S') = 0$, $C(v) = +\infty$; if there exists $S \in \bar{S}_o(v)$ such that $A(S) = 0$ and no $S' \in \bar{S}_o(v)$ such that $A(S') = 1$, $C(v) = -\infty$.*

Note that the first property is actually a justification for the naive voting strategy when all sources have the same accuracy. The third property shows that we should be careful not to assign very high or very low accuracy to a data source, which has been avoided by defining the accuracy of a source as the average probability of its provided values.

Example 2. Consider S_1, S_2 and S_3 in Table 1 and assume their accuracies are .97, .6, .4 respectively. Assuming there are 5 false values in the domain (*i.e.*, $n = 5$), we can compute the accuracy score of each source as follows. For S_1 , $A'(S_1) = \log \frac{5*.97}{1-.97} = 4.7$; for S_2 , $A'(S_2) = \log \frac{5*.6}{1-.6} = 2$; and for S_3 , $A'(S_3) = \log \frac{5*.4}{1-.4} = 1.5$.

Now consider the three values provided for *Carey*. Value *UCI* thus has confidence 8, *AT&T* has confidence 5, and *BEA* has confidence 4. Among them, *UCI* has the highest confidence and so the highest probability to be true. Indeed, its probability is $\frac{2^8}{2^8+2^5+2^4+(5-2)*2^0} = .9$.

Computing value confidence requires knowing accuracy of data sources, whereas computing source accuracy requires knowing value probability. There is an inter-dependence between them and we solve the problem by computing them iteratively. We give details of the iterative algorithm in Section 3.

3 Fusing Sources Considering Copying

Next, we describe how we detect copiers and leverage the discovered copying relationships in data fusion.

3.1 Copy Detection

We say that there exists *copying* between two data sources S_1 and S_2 if they derive the same part of their data directly or transitively from a common source (can be one of S_1 and S_2). Accordingly, there are two types of data sources: *independent sources* and *copiers*. An *independent source* provides all values independently. It may provide some erroneous values because of incorrect knowledge of the real world, mis-spellings, etc. A *copier* copies a part (or all) of its data from other sources (independent sources or copiers). It can copy from multiple sources by union, intersection, etc., and as we focus on a snapshot of data, cyclic copying on a particular object is impossible. In addition, a copier may revise some of the copied values or add additional values; though, such revised and added values are considered as independent contributions of the copier.

To make our models tractable, we consider only *direct* copying. In addition, we make the following assumptions.

- *Assumption 1 (Independent values)*. The values that are independently provided by a data source on different objects are independent of each other.
- *Assumption 2 (Independent copying)*. The copying between a pair of data sources is independent of the copying between any other pair of data sources.
- *Assumption 3 (No mutual copying)*. There is no mutual copying between a pair of sources; that is, S_1 copying from S_2 and S_2 copying from S_1 do not happen at the same time.

Our experiments on real world data show that the basic model already obtains high accuracy and we refer our readers to [6] for how we can relax the assumptions. We next describe the basic copy-detection model.

Consider two sources $S_1, S_2 \in \mathcal{S}$. We apply Bayesian analysis to compute the probability of copying between S_1 and S_2 given observation of their data. For this purpose, we need to compute the probability of the observed data, conditioned on independence of or copying between the sources. We denote by c ($0 < c \leq 1$) the probability that a value provided by a copier is copied. We bootstrap our algorithm by setting c to a default value initially and iteratively refine it according to copy detection results.

In our observation, we are interested in three sets of objects: \bar{O}_t , denoting the set of objects on which S_1 and S_2 provide the same true value, \bar{O}_f , denoting the set of objects on which they provide the same false value, and \bar{O}_d , denoting the set of objects on which they provide different values ($\bar{O}_t \cup \bar{O}_f \cup \bar{O}_d \subseteq \mathcal{O}$). Intuitively, two independent sources providing the same false value is a low-probability event; thus, if we fix $\bar{O}_t \cup \bar{O}_f$ and \bar{O}_d , the more common false values that S_1 and S_2 provide, the more likely that they are dependent. On the other hand, if we fix \bar{O}_t and \bar{O}_f , the fewer objects on which S_1 and S_2 provide different values, the more likely that they are dependent. We denote by Φ the observation of $\bar{O}_t, \bar{O}_f, \bar{O}_d$ and by k_t, k_f and k_d their sizes respectively. We next describe how we compute the conditional probability of Φ based on these intuitions.

We first consider the case where S_1 and S_2 are independent, denoted by $S_1 \perp S_2$. Since there is a single true value, the probability that S_1 and S_2 provide the same true value for object O is

$$\Pr(O \in \bar{O}_t | S_1 \perp S_2) = A(S_1) \cdot A(S_2). \quad (5)$$

On the other hand, the probability that S_1 and S_2 provide the same false value for O is

$$Pr(O \in \bar{O}_f | S_1 \perp S_2) = n \cdot \frac{1 - A(S_1)}{n} \cdot \frac{1 - A(S_2)}{n} = \frac{(1 - A(S_1))(1 - A(S_2))}{n}. \quad (6)$$

Then, the probability that S_1 and S_2 provide different values on an object O , denoted by P_d for convenience, is

$$Pr(O \in \bar{O}_d | S_1 \perp S_2) = 1 - A(S_1)A(S_2) - \frac{(1 - A(S_1))(1 - A(S_2))}{n} = P_d. \quad (7)$$

Following the *Independent-values* assumption, the conditional probability of observing Φ is

$$Pr(\Phi | S_1 \perp S_2) = \frac{A(S_1)^{k_t} A(S_2)^{k_t} (1 - A(S_1))^{k_f} (1 - A(S_2))^{k_f} P_d^{k_d}}{n^{k_f}}. \quad (8)$$

We next consider the case when S_2 copies from S_1 , denoted by $S_2 \rightarrow S_1$. There are two cases where S_1 and S_2 provide the same value v for an object O . First, with probability c , S_2 copies v from S_1 and so v is true with probability $A(S_1)$ and false with probability $1 - A(S_1)$. Second, with probability $1 - c$, the two sources provide v independently and so its probability of being true or false is the same as in the case where S_1 and S_2 are independent. Thus, we have

$$Pr(O \in \bar{O}_t | S_2 \rightarrow S_1) = A(S_1) \cdot c + A(S_1) \cdot A(S_2) \cdot (1 - c), \quad (9)$$

$$Pr(O \in \bar{O}_f | S_2 \rightarrow S_1) = (1 - A(S_1)) \cdot c + \frac{(1 - A(S_1))(1 - A(S_2))}{n} \cdot (1 - c). \quad (10)$$

Finally, the probability that S_1 and S_2 provide different values on an object is that of S_1 providing a value independently and the value differs from that provided by S_2 :

$$Pr(O \in \bar{O}_d | S_2 \rightarrow S_1) = P_d \cdot (1 - c). \quad (11)$$

We compute $Pr(\Phi | S_2 \rightarrow S_1)$ accordingly; similarly we can also compute $Pr(\Phi | S_1 \rightarrow S_2)$. Now we can compute the probability of $S_1 \perp S_2$ by applying the Bayes Rule.

$$\begin{aligned} & Pr(S_1 \perp S_2 | \Phi) \\ &= \frac{\alpha Pr(\Phi | S_1 \perp S_2)}{\alpha Pr(\Phi | S_1 \perp S_2) + \frac{1-\alpha}{2} Pr(\Phi | S_1 \rightarrow S_2) + \frac{1-\alpha}{2} Pr(\Phi | S_2 \rightarrow S_1)}. \end{aligned} \quad (12)$$

Here $\alpha = Pr(S_1 \perp S_2)$ ($0 < \alpha < 1$) is the *a priori* probability that two data sources are independent. As we have no *a priori* preference for copy direction, we set the *a priori* probability for copying in each direction as $\frac{1-\alpha}{2}$.

Equation (12) has several nice properties that conform to the intuitions we discussed earlier in this section, formalized as follows.

Theorem 2. *Let \mathcal{S} be a set of good independent sources and copiers. Equation (12) has the following three properties on \mathcal{S} .*

1. Fixing $k_t + k_f$ and k_d , when k_f increases, the probability of copying (i.e., $\Pr(S_1 \rightarrow S_2 | \Phi) + \Pr(S_2 \rightarrow S_1 | \Phi)$) increases;
2. Fixing $k_t + k_f + k_d$, when $k_t + k_f$ increases and none of k_t and k_f decreases, the probability of copying increases;
3. Fixing k_t and k_f , when k_d decreases, the probability of copying increases.

Example 3. Continue with Ex.1 and consider the possible copying relationship between S_1 and S_2 . We observe that they share no false values (all values they share are correct), so copying is unlikely. With $\alpha = .5, c = .2, A(S_1) = .97, A(S_2) = .6$, the Bayesian analysis goes as follows.

We start with computation of $\Pr(\Phi | S_1 \perp S_2)$. We have $\Pr(O \in \bar{O}_t | S_1 \perp S_2) = .97 * .6 = .582$. There is no object in \bar{O}_f and we denote by P_d the probability $\Pr(O \in \bar{O}_f | S_1 \perp S_2)$. Thus, $\Pr(\Phi | S_1 \perp S_2) = .582^3 * P_d^2 = .2P_d^2$.

Next consider $\Pr(\Phi | S_1 \rightarrow S_2)$. We have $\Pr(O \in \bar{O}_t | S_1 \rightarrow S_2) = .8 * .6 + .2 * .582 = .6$ and $\Pr(O \in \bar{O}_f | S_1 \rightarrow S_2) = .2P_d$. Thus, $\Pr(\Phi | S_1 \rightarrow S_2) = .6^3 * (.2P_d)^2 = .008P_d^2$. Similarly, $\Pr(\Phi | S_2 \rightarrow S_1) = .028P_d^2$.

According to Equation (12), $\Pr(S_1 \perp S_2 | \Phi) = \frac{.5 * .2P_d^2}{.5 * .2P_d^2 + .25 * .008P_d^2 + .25 * .028P_d^2} = .92$, so independence is very likely.

3.2 Independent Vote Count of a Value

Since even a copier can provide some of the values independently, we compute the *independent* vote for each particular value. In this process we consider the data sources one by one in some order. For each source S , we denote by $\overline{\text{Pre}}(S)$ the set of sources that have already been considered and by $\overline{\text{Post}}(S)$ the set of sources that have not been considered yet. We compute the probability that the value provided by S is independent of any source in $\overline{\text{Pre}}(S)$ and take it as the vote count of S . The vote count computed in this way is not precise because if S depends only on sources in $\overline{\text{Post}}(S)$ but some of those sources depend on sources in $\overline{\text{Pre}}(S)$, our estimation still (incorrectly) counts S 's vote. To minimize such error, we wish that the probability that S depends on a source $S' \in \overline{\text{Post}}(S)$ and S' depends on a source $S'' \in \overline{\text{Pre}}(S)$ be the lowest. Thus, we use a greedy algorithm and consider data sources in the following order.

1. If the probability of $S_1 \rightarrow S_2$ is much higher than that of $S_2 \rightarrow S_1$, we consider S_1 as a copier of S_2 with probability $\Pr(S_1 \rightarrow S_2 | \Phi) + \Pr(S_2 \rightarrow S_1 | \Phi)$ (recall that we assume there is no mutual-copying) and order S_2 before S_1 . Otherwise, we consider both directions as equally possible and there is no particular order between S_1 and S_2 ; we consider such copying *undirectional*.
2. For each subset of sources between which there is no particular ordering yet, we sort them as follows: in the first round, we select a data source that is associated with the undirectional copying of the highest probability ($\Pr(S_1 \rightarrow S_2 | \Phi) + \Pr(S_2 \rightarrow S_1 | \Phi)$); in later rounds, each time we select a data source that has the copying with the maximum probability with one of the previously selected sources.

We now consider how to compute the vote count of v once we have decided an order of the data sources. Let S be a data source that votes for v . The probability that S provides

v independently of a source $S_0 \in \overline{Pre}(S)$ is $1 - c(Pr(S_1 \rightarrow S_0 | \Phi) + Pr(S_0 \rightarrow S_1 | \Phi))$ and the probability that S provides v independently of any data source in $\overline{Pre}(S)$, denoted by $I(S)$, is

$$I(S) = \prod_{S_0 \in \overline{Pre}(S)} (1 - c(Pr(S_1 \rightarrow S_0 | \Phi) + Pr(S_0 \rightarrow S_1 | \Phi))). \quad (13)$$

The total vote count of v is $\sum_{S \in \bar{S}_o(v)} I(S)$.

Finally, when we consider the accuracy of sources, we compute the confidence of v as follows.

$$C(v) = \sum_{S \in \bar{S}_o(v)} A'(S)I(S). \quad (14)$$

In the equation, $I(S)$ is computed by Equation (13). In other words, we take only the “independent fraction” of the original vote count (decided by source accuracy) from each source.

3.3 Iterative Algorithm

We need to compute three measures: accuracy of sources, copying between sources, and confidence of values. Accuracy of a source depends on confidence of values; copying between sources depends on accuracy of sources and the true values selected according to the confidence of values; and confidence of values depends on both accuracy of and copying between data sources.

We conduct analysis of both accuracy and copying in each round. Specifically, Algorithm ACCUCOPY starts by setting the same accuracy for each source and the same probability for each value, then iteratively (1) computes copying based on the confidence of values computed in the previous round, (2) updates confidence of values accordingly, and (3) updates accuracy of sources accordingly, and stops when the accuracy of the sources becomes stable. Note that it is crucial to consider copying between sources from the beginning; otherwise, a data source that has been duplicated many times can dominate the voting results in the first round and make it hard to detect the copying between it and its copiers (as they share only “true” values). Our initial decision on copying is similar to Equation (12) except considering both the possibility of a value being true and that of the value being false and we skip details here.

We can prove that if we ignore source accuracy (*i.e.*, assuming all sources have the same accuracy) and there are a finite number of objects in \mathcal{O} , Algorithm ACCUCOPY cannot change the decision for an object O back and forth between two different values forever; thus, the algorithm converges.

Theorem 3. *Let \mathcal{S} be a set of good independent sources and copiers that provide information on objects in \mathcal{O} . Let l be the number of objects in \mathcal{O} and n_0 be the maximum number of values provided for an object by \mathcal{S} . The ACCUVOTE algorithm converges in at most $2ln_0$ rounds on \mathcal{S} and \mathcal{O} if it ignores source accuracy.*

Once we consider accuracy of sources, ACCUCOPY may not converge: when we select different values as the true values, the direction of the copying between two sources can change and in turn suggest different true values. We stop the process after we detect

oscillation of decided true values. Finally, we note that the complexity of each round is $O(|\mathcal{O}||\mathcal{S}|^2 \log |\mathcal{S}|)$.

4 A Case Study

We now describe a case study on a real-world data set¹ extracted by searching computer-science books on *AbeBooks.com*. For each book, *AbeBooks.com* returns information provided by a set of online bookstores. Our goal is to find the list of authors for each book. In the data set there are 877 bookstores, 1263 books, and 24364 listings (each listing contains a list of authors on a book provided by a bookstore).

We did a normalization of author names and generated a normalized form that preserves the order of the authors and the first name and last name (ignoring the middle name) of each author. On average, each book has 19 listings; the number of different author lists after cleaning varies from 1 to 23 and is 4 on average.

Table 2. Different types of errors by naive voting

Missing authors	Additional authors	Mis-ordering	Mis-spelling	Incomplete names
23	4	3	2	2

Table 3. Results on the book data set. For each method, we report the precision of the results, the run time, and the number of rounds for convergence. ACCUCOPY and COPY obtain a high precision.

Model	Precision	Rounds	Time (sec)
VOTE	.71	1	.2
SIM	.74	1	.2
ACCU	.79	23	1.1
COPY	.83	3	28.3
ACCUCOPY	.87	22	185.8
ACCUCOPYSIM	.89	18	197.5

We used a golden standard that contains 100 randomly selected books and the list of authors found on the cover of each book. We compared the fusion results with the golden standard, considering missing or additional authors, mis-ordering, misspelling, and missing first name or last name as errors; however, we do not report missing or misspelled middle names. Table 2 shows the number of errors of different types on the selected books if we apply a naive voting (note that the result author lists on some books may contain multiple types of errors).

We define *precision* of the results as the fraction of objects on which we select the true values (as the number of true values we return and the real number of true values

¹ <http://lunadong.com/fusionDataSets.htm>

are both the same as the number of objects, the *recall* of the results is the same as the precision). Note that this definition is different from that of accuracy of sources.

Precision and Efficiency. We compared the following data fusion models on this data set.

- VOTE conducts naive voting;
- SIM conducts naive voting but considers similarity between values;
- ACCU considers accuracy of sources as we described in Section 2, but assumes all sources are independent;
- COPY considers copying between sources as we described in Section 3, but assumes all sources have the same accuracy;
- ACCUCOPY applies the ACCUCOPY algorithm described in Section 3, considering both source accuracy and copying.
- ACCUCOPYSIM applies the ACCUCOPY algorithm and considers in addition similarity between values.

When applicable, we set $\alpha = .2, c = .8, \varepsilon = .2$ and $n = 100$. Though, we observed that ranging α from .05 to .5, ranging c from .5 to .95, and ranging ε from .05 to .3 did not change the results much. We compared similarity of two author lists using 2-gram Jaccard distance.

Table 3 lists the precision of results of each algorithm. ACCUCOPYSIM obtained the best results and improved over VOTE by 25.4%. SIM, ACCU and COPY each extends

Table 4. Bookstores that are likely to be copied by more than 10 other bookstores. For each bookstore we show the number of books it lists and its accuracy computed by ACCUCOPYSIM.

Bookstore	#Copiers	#Books	Accuracy
Caiman	17.5	1024	.55
MildredsBooks	14.5	123	.88
COBU GmbH & Co. KG	13.5	131	.91
THESAINTBOOKSTORE	13.5	321	.84
Limelight Bookshop	12	921	.54
Revaluation Books	12	1091	.76
Players Quest	11.5	212	.82
AshleyJohnson	11.5	77	.79
Powell's Books	11	547	.55
AlphaCraze.com	10.5	157	.85
Avg	12.8	460	.75

Table 5. Difference between accuracy of sources computed by our algorithms and the sampled accuracy on the golden standard. The accuracy computed by ACCUCOPYSIM is the closest to the sampled accuracy.

	Sampled	ACCUCOPYSIM	ACCUCOPY	ACCU
Average source accuracy	.542	.607	.614	.623
Average difference	-	.082	.087	.096

VOTE on a different aspect; while each of them increased the precision, COPY increased it the most.

To further understand how considering copying and accuracy of sources can affect our results, we looked at the books on which ACCUCOPY and VOTE generated different results and manually found the correct authors. There are 143 such books, among which ACCUCOPY gave correct authors for 119 books, VOTE gave correct authors for 15 books, and both gave incorrect authors for 9 books.

Finally, COPY was quite efficient and finished in 28.3 seconds. It took ACCUCOPY and ACCUCOPYSIM longer time to converge (3.1, 3.3 minutes respectively); however, truth discovery is often a one-time process and so taking a few minutes is reasonable.

Copying and Source Accuracy: Out of the 385,000 pairs of bookstores, 2916 pairs provide information on at least the same 10 books and among them ACCUCOPYSIM found 508 pairs that are likely to be dependent. Among each such pair S_1 and S_2 , if the probability of S_1 depending on S_2 is over 2/3 of the probability of S_1 and S_2 being dependent, we consider S_1 as a *copier* of S_2 ; otherwise, we consider S_1 and S_2 each has .5 probability to be a *copier*. Table 4 shows the bookstores whose information is likely to be copied by more than 10 bookstores. On average each of them provides information on 460 books and has accuracy .75. Note that among all bookstores, on average each provides information on 28 books, conforming to the intuition that small bookstores are more likely to copy data from large ones. Interestingly, when we applied VOTE on only the information provided by bookstores in Table 4, we obtained a precision of only .58, showing that bookstores that are large and copied often actually can make a lot of mistakes.

Finally, we compare the source accuracy computed by our algorithms with that sampled on the 100 books in the golden standard. Specifically, there were 46 bookstores that provide information on more than 10 books in the golden standard. For each of them we computed the *sampled accuracy* as the fraction of the books on which the bookstore provides the same author list as the golden standard. Then, for each bookstore we computed the difference between its accuracy computed by one of our algorithms and the sampled accuracy (Table 5). The source accuracy computed by ACCUCOPYSIM is the closest to the sampled accuracy, indicating the effectiveness of our model on computing source accuracy and showing that considering copying between sources helps obtain better source accuracy.

5 Related Work and Conclusions

This paper presented how to improve truth discovery by analyzing accuracy of sources and detecting copying between sources. We describe Bayesian models that discover copiers by analyzing values shared between sources. A case study shows that the presented algorithms can significantly improve accuracy of truth discovery and are scalable when there are a large number of data sources.

Our work is closely related to *Data Provenance*, which has been a topic of research for a decade [4, 5]. Whereas research on data provenance is focused on how to represent and analyze available provenance information, our work on copy detection helps detect provenance and in particular copying relationships between dependent data sources.

Our work is also related to analysis of trust and authoritativeness of sources [1–3, 10, 9, 12] by link analysis or source behavior in a P2P network. Such trustworthiness is not directly related to source accuracy.

Finally, various fusion models have been proposed in the literature. A comparison of them is presented in [11] on two real-world Deep Web data sets, showing advantages of considering source accuracy together with copying in data fusion.

References

1. Artz, D., Gil, Y.: A survey of trust in computer science and the semantic web. *Journal of Web Semantics* 5(2) (2010)
2. Borodin, A., Roberts, G., Rosenthal, J., Tsaparas, P.: Link analysis ranking: algorithms, theory, and experiments. *TOIT* 5, 231–297 (2005)
3. Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30(1-7), 107–117 (1998)
4. Buneman, P., Cheney, J., Tan, W.-C., Vansummeren, S.: Curated databases. In: Proc. of PODS (2008)
5. Davidson, S., Freire, J.: Provenance and scientific workflows: Challenges and opportunities. In: Proc. of SIGMOD (2008)
6. Dong, X.L., Berti-Equille, L., Hu, Y., Srivastava, D.: Global detection of complex copying relationships between sources. *PVLDB* (2010)
7. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: the role of source dependence. *PVLDB* 2(1) (2009)
8. Dong, X.L., Berti-Equille, L., Srivastava, D.: Truth discovery and copying detection in a dynamic world. *PVLDB* 2(1) (2009)
9. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The Eigentrust algorithm for reputation management in P2P networks. In: WWW (2003)
10. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: SODA (1998)
11. Li, X., Dong, X.L., Lyons, K.B., Meng, W., Srivastava, D.: Truth finding on the deep web: Is the problem solved? *PVLDB* 6(2) (2013)
12. Singh, A., Liu, L.: TrustMe: anonymous management of trust relationships in decentralized P2P systems. In: IEEE Intl. Conf. on Peer-to-Peer Computing (2003)
13. Zhao, B., Rubinstein, B.I.P., Gemmell, J., Han, J.: A bayesian approach to discovering truth from conflicting sources for data integration. *PVLDB* 5(6), 550–561 (2012)

Entity Resolution on Uncertain Relations

Huabin Feng, Hongzhi Wang, Jianzhong Li, and Hong Gao

Harbin Institute of Technology

fenghuabin@gmail.com, {whongzhi,lijzh,honggao}@hit.edu.cn

Abstract. In many different application areas entity resolution places a pivotal role. Because of the existence of uncertain in many applications such as information extraction and online product category, entity resolution should be applied on uncertain data. The characteristic of uncertainty makes it impossible to apply traditional techniques directly. In this paper, we propose techniques to perform entity resolution on uncertain data. Firstly, we propose a new probabilistic similarity metric for uncertain tuples. Secondly, based on the metric, we propose novel pruning techniques to efficiently join pairwise uncertain tuples without enumerating all possible worlds. Finally, we propose a density-based clustering algorithm to combine the results of pairwise similarity join. With extensive experimental evaluation on synthetic and real-world data sets, we demonstrate the benefits and features of our approaches.

1 Introduction

Uncertainty is common in data generated and transmitted in sensor networks or integrated from different databases. Besides, data mapping in uncertain data integrating environments or data in some specific applications involving privacy and confidentiality often contain uncertainty. The case that each real-world entity has multiple representations often occurs on uncertain data.

For a B2C website, a product may have many alias names and there may be many merchants offering the product with different alias names and accordingly, with different descriptions. Similarly, although a kind of drug only has one chemical name, it may have various kinds of alias names (e.g., common names, commercial names) along with different specifications. So the recognition of different representations for the same real-world entity on uncertain data sets is in demand.

The task of finding records referring to the same real-world entity is called entity resolution. It is particularly important in data cleaning and data integration. To resolve entities on deterministic datasets, a common framework is to join pairwise relations and then combine these pairwise join results to analyze and partition all records into groups with each of them referring to the same real-world entity.

The basic technique for joining deterministic relations is to perform the threshold-based set similarity join under some set similarity function (e.g., Jaccard similarity, Cosine similarity, and overlap similarity). All traditional similarity join algorithms take the exact similarity metrics as input and the similarity metrics are designed only for deterministic relations. Hence we cannot adapt these techniques

to uncertain tuples without similarity metrics and correspondent join algorithms available.

To solve this problem, we firstly propose models for uncertain relations. Then, we propose a probabilistic similarity metric and a corresponding pairwise join algorithm. To accelerate the join process, we design novel pruning techniques. Finally, we propose a density-based clustering algorithm to combine the pairwise results.

In this paper, we first state the problem definition in section 2. Then in Section 3, we deal with pairwise entity resolution. In Section 4, we state our group-wise entity resolution algorithm. Finally, the experimental evaluation is given in Section 5.

2 Problem Definition

To model uncertain relations, a natural way is to list all the possible instances of an uncertain tuple with their corresponding probability within a set. Compared to the traditional possible world model, the benefit of this model is that it can describe any kinds of probability distribution functions of uncertain relations visually and also it can be stored in relational databases directly. So we define two new probabilistic models in two levels, e.g., tuple-level and attribute-level as follows.

Definition 1 (tuple-level model). Let Σ be a set of all possible tuples. A tuple-level uncertain relation is $S = \{(\sigma_1, p_1), \dots, (\sigma_m, p_m)\}$ where $\sigma_i \in \Sigma$, $p_i \in (0, 1]$ and $\sum_{i=1}^m p_i = 1$. S instantiates into σ_i with probability p_i independently. We use $S(i)$ to denote i -th choice, i.e., $S(i) = (\sigma_i, p_i)$ for $i = 1 \dots m$.

The tuple-level model can represent any form of uncertain relations. However, the uncertainty of uncertain tuples often exists only at a few locations within certain attributes along with several other deterministic attributes. Representing such uncertain tuples in the tuple-level model would be highly redundant. To avoid this, we propose the attribute-level model.

Definition 2 (attribute-level model). Let $\Sigma_1 \dots \Sigma_n$ be the n possible sets of the n attributes of an uncertain tuple. An attribute-level uncertain tuple is $S = S[1] \dots S[n]$. For $i = 1, \dots, n$, $S[i] = \{(\eta_{i,1}, p_{i,1}), \dots, (\eta_{i,\theta_i}, p_{i,\theta_i})\}$, where $\eta_{i,j} \in \Sigma_i$, all the attribute instances $\eta_{i,j}$ are mutually exclusive; each instance $\eta_{i,j}$ has an existence probability $p_{i,j} \in (0, 1]$ and $\sum_{j=1}^{\theta_i} p_{i,j} = 1$, that is saying each $S[i]$ instantiates into $\eta_{i,j}$ with probability $p_{i,j}$ independently.

The attribute-level model allows us to represent exponentially many possible instances of an uncertain tuple. If there is non-negligible correlation between adjacent attributes, these two models could be combined by making each $S[i]$ a tuple-level uncertain tuple instead of one single attribute. Based on the above models, we define the problem of entity resolution on uncertain relations.

Definition 3 (Entity Resolution on Uncertain Relations) Given a table T of N uncertain tuples $T = \{T_1, T_2, \dots, T_N\}$, the problem of Entity Resolution on Uncertain Relations is to seek a K -partition of T , $C = \{C_1, C_2, \dots, C_K\}$ ($K \leq N$), such that a) $C_i \neq \emptyset$, $i = 1, \dots, N$; b) $\bigcup_{i=1}^N C_i = T$; c) $C_i \cap C_j = \emptyset$, $i \neq j$ and $i, j = 1, \dots, N$. It

means that all the uncertain tuples in the same cluster C_i refer to the same real-world entity and tuples in the different clusters referring to the different entities.

This problem could be solved by firstly conducting pairwise uncertain entity resolution and then aggregating the pairwise results. As the string type relation is dormant in real world, the following techniques are designed for uncertain strings of single attribute in attribute-level model or multi-attributes in tuple-level model.

To perform the first step, we propose a probabilistic metric that considers the *possible world* semantics, where each possible world is a materialized instance of pairwise strings that can occur in the real world. A possible world s is a pair $\{(\sigma_1, p_1), (\sigma_2, p_2)\}$, where $\sigma_1(\sigma_2)$ is an instance from uncertain strings $S_1(S_2)$ with probability $p_1(p_2)$. The probability of s is $p(s) = p_1 * p_2$.

3 Pair-Wise Entity Resolution

As is stated before, similarity join on uncertain strings is the key step of pairwise entity resolution. Based on *possible worlds* semantics, we introduce a probabilistic similarity metric $Pr\{sim(\sigma_i, \sigma_j) \geq \tau\} = \sum\{p(\sigma_i, \sigma_j) | sim(\sigma_i, \sigma_j) \geq \tau\}$, which accumulates the probability of possible worlds with similarity above a threshold τ . Based on this metric, given a similarity threshold $\tau \in (0,1]$ and a probability threshold $\alpha \in (0,1]$, we can join uncertain strings S_1 and S_2 by the following:

$$Pr\{sim(\sigma_i, \sigma_j) \geq \tau\} \geq \alpha \quad (1)$$

A trivial method of computing $Pr\{sim(\sigma_i, \sigma_j) \geq \tau\}$ is to retrieval all the possible worlds in a nested loop manner which incurs $O(|S_1| \cdot |S_2|)$ complexity, where $|S_i|$ denotes number of choices in S_i . Clearly, such approach is inefficient, especially when there is exponential number of possible worlds. Thus we need to propose new pruning algorithm to efficiently join uncertain strings.

In this section, we firstly provide backgrounds for the prefix and positional filtering principles and several related optimizations designed for deterministic strings, and then we extend these techniques for uncertain tuples and present our probabilistic similarity join algorithm.

3.1 Background

For our similarity metric, we focus on a class of set similarity functions (e.g., Jaccard, Cosine, Dice, or Overlap similarity) by representing string as a set of features in the concept of q -grams. Nonetheless, the above metrics are all inter-related and can be converted into overlap similarity [1,7,9,5,8]. So, we will focus on Jaccard similarity. We do not distinguish a string σ and its corresponding q -gram set in the following.

The Jaccard similarity constraint $J(\sigma_1, \sigma_2) = \tau$ can be transformed into a lower overlap similarity constraint [4]:

$$O(\sigma_1, \sigma_2) = |\sigma_1 \cap \sigma_2| \geq \alpha = \tau(|\sigma_1| + |\sigma_2|)/(1 + \tau) \quad (2)$$

By carefully exploiting definition of Jaccard similarity, we can infer the *size bounds*, e.g., any sets whose sizes are not within $[\tau|\sigma|, |\sigma|/\tau]$ can be safely pruned.

Overlap and set size bounds give other optimization chances. The *prefix filtering* principles ([2][4][9]) (stated in Lemma 1) is based on the observation that if two strings are similar, they will probably share a large percent of common fragments.

Lemma 1 (Prefix Filtering Principle). *Let the p -prefix of a string σ be the first p q -grams of σ . Let $O(\sigma_1, \sigma_2) = |\sigma_1 \cap \sigma_2|$ denote the overlap similarity of two string, if $O(\sigma_1, \sigma_2) \geq \alpha$, then the $(|\sigma_1| - \alpha + 1)$ -prefix of σ_1 and the $(|\sigma_2| - \alpha + 1)$ -prefix of σ_2 must share at least one q -gram.*

An efficient implementation of the prefix filtering principle is to construct an inverted index on features of p -prefix. In order to identify all candidates of σ , according the *size bounds*, we need a prefix of length $|\sigma| - [\tau \cdot |\sigma|] + 1$ for each string σ to construct the inverted lists. Besides, the ordering of grams can be explored to improve the pruning efficiency as is stated in Lemma 2 [9].

Lemma 2 (Positional Filtering Principle). *Consider an ordering \mathcal{O} of the grams universe U and a set of strings, each with grams sorted by \mathcal{O} . Let gram $f = \sigma[i]$, f partitions the string into the left partition $\sigma_l(f) = \sigma[1..(i-1)]$ and the right partition $\sigma_r(f) = \sigma[i..|\sigma|]$. If $O(\sigma, \sigma^*) \geq \alpha$, then for every token $f \in \sigma \cap \sigma^*$, $O(\sigma_l(f), \sigma^{*l}(f)) + \min(|\sigma_r(f)|, |\sigma^{*r}(f)|) \geq \alpha$.*

3.2 Adaptation for Uncertain Relations

In the *possible worlds* semantics, we can apply the above techniques to join uncertain strings directly. Not that the above techniques treat all grams equally. But, in practice, when uncertain strings are combined by multiple attributes, grams from different attributes should be treated differently. As different attributes have different importance in distinguish the overall strings from others. So it is in need for considering weighted grams and developing weighted versions of the above techniques.

In weighted version, all the grams in a dataset form a universe U and each gram f from the universe is related with a weight $w(f)$. Thus we can compute the weighted size of a string σ , e.g., $w(\sigma) = \sum_f w(f)$.

Then given a weighted overlap lower bound α , the weighted prefix of a set σ ($\text{pref}(\sigma)$) is the shortest subset of σ such that $w(\text{pref}(\sigma)) > w(\sigma) - \alpha$.

The weighted Jaccard similarity constraint $wJ(\sigma_1, \sigma_2) \geq \tau$ can be transformed into corresponding weighted lower overlap bound:

$$wO(\sigma_1, \sigma_2) \geq \alpha = \tau(w(\sigma_1) + w(\sigma_2))/(1 + \tau) \quad (3)$$

Also, the corresponding weighted version of *size bounds* is $[\tau w(\sigma_1), w(\sigma_1)/\tau]$. From the weighted lower overlap and size bounds, we have $\alpha \geq \tau w(\sigma_1)$. So, to ensure the weighted version does not miss any candidates, the probing prefix length of σ_1 is given next.

$$l_p = \min \{j | w(\sigma_1) - \sum_{i=1}^j w(\sigma_1[i]) < \tau w(\sigma_1)\} \quad (4)$$

Algorithm 1. The *wpro-PJOIN* algorithm

Input: uncertain strings R_1 and R_2 ; thresholds τ and α

Output: The join result

```

1   $R = R_1 \cup R_2$ , sorted in ascending order strings' weights
2   $T_\sigma = 1$  when  $\sigma \in R_1$ ,  $T_\sigma = 2$  when  $\sigma \in R_2$ ,  $I_f \leftarrow \emptyset$ ,  $Pr_f, Pr_t \leftarrow 0$ 
3  for each  $\sigma \in R$  do
4       $A[|R|] \leftarrow 0$ , vector of accumulated overlap;
5       $l_p$  is calculate from Eq. (4);
6      for  $i = 1$  to  $l_p$  do
7           $f \leftarrow \sigma[i]$ ;
8          for each  $(\sigma^*, j) \in I_f$  such that  $w(\sigma^*) \geq \tau * w(\sigma) \wedge T_\sigma \neq T_{\sigma^*}$  do
9               $\gamma = \left\lceil \frac{\tau}{1+\tau} (w(\sigma) + w(\sigma^*)) \right\rceil$ ;
10              $ubound \leftarrow \min(w_\sigma[i..|\sigma|], w_{\sigma^*}[j..|\sigma^*|])$ ;
11             if  $A[\sigma^*] + ubound \geq \gamma$  then
12                  $A[\sigma^*] = A[\sigma^*] + w(f)$ ;
13             else
14                  $A[\sigma^*] \leftarrow -1$ ;
15                 FalseAddTest( $Pr_f, Pr(\sigma, \sigma^*)$ );
16              $I_f \leftarrow I_f \cup \{(\sigma, w_\sigma[i..|\sigma|])\}$ ;
17             for each  $\sigma^* \in R$  such that  $O(\sigma^*) > O(\sigma)$  and  $A[\sigma^*] = 0$ 
18                 FalseAddTest( $Pr_f, Pr(\sigma, \sigma^*)$ );
19             for each  $\sigma^*$  such that  $A[\sigma^*] > 0$  do
20                  $S \leftarrow S \cup \{(\sigma, \sigma^*)\}$ ;
21 sort  $S$  in decreasing order of each possible world's probability
22 for each  $(\sigma, \sigma^*) \in S$ 
23     if Verify( $\sigma, \sigma^*, \alpha$ )
24         TrueAddTest( $Pr_t, Pr(\sigma, \sigma^*)$ );
25     else
26         FalseAddTest( $Pr_f, Pr(\sigma, \sigma^*)$ );
27 return;
```

Based on the above weighted optimizations and inspired by ppjoin [9], we designed *wpro-PJOIN* (Algorithm 1) to join uncertain strings. It takes input a pair of uncertain strings R_1 and R_2 , along with the similarity and probability thresholds. Each string's set are sorted in the increasing order of each grams' weights.

The algorithm contains the candidate generation phase and the verification phase. In the candidate generation phase, the algorithm looks for candidates that intersect with σ 's weighted probing prefix l_p (calculated by Eq.(4)) for each string σ in R . All the possible worlds surviving the prefix and positional filtering principles are collected in S . The integration of positional filtering principle incurs the computation of the lower overlap constraint γ of weighted Jaccard similarity threshold (Line 9) and the weighted upper bound of the overlap $ubound$ between right partitions of σ and σ^* w.r.t the current feature f (Line 10). The algorithm marks a string if it is pruned by positional filtering principle (Line 14). For the weighted version of

positional filtering principle, we index the sum of the weights of unseen grams in the inverted list as the “positional information”.

Next is the verification phase, where possible worlds in S will be verified to involve probability filtering principle. Specifically, when a possible world is pruned or verified conflict with the similarity prediction of Eq.(1), the procedure *FalseAddTest* will be called to add its probability to *false probability* (Pr_f) and compare Pr_f with $1 - \alpha$. If Pr_f reaches $1 - \alpha$, *wpro-PJOIN* will stop and returns “dissimilar” as join results. While *TrueAddTest* acts inversely. We sort possible worlds in S by the decreasing order of their probability (Line 21) to guarantee the smallest number of verifications before the probability filtering principle reached its pruning thresholds.

Note that, we only need to consider pairwise instances from different uncertain tuples, i.e., $T_\sigma \neq T_{\sigma^*}$ (Line 8). The algorithm also employs weighted *size bound* to reduce accesses to inverted lists (Line 8). The time complexity of the algorithm is $O(\alpha \cdot |S_1| \cdot |S_2|)$ w.r.t the uniform distribution of possible worlds.

4 Group-Wise Entity Resolution on Uncertain Relations

In this section, we discuss group-wise entity resolution on uncertain relations. The results of pairwise join results are represented by an undirected graph $G = (V, E)$, where an edge (v_i, v_j) in E exists if Eq.(1) holds for uncertain tuples represented by v_i and v_j . The goal is to partition vertices in G into clusters such that vertices in each partition refer to the same entity.

Without extra parameters such as the number of clusters, we attempt to use a density-based clustering. It considers high-density areas according to a density metric as a cluster and does not require extra parameters. Given distance metric dis and a radius defined on this metric, the key idea of density-based clustering is that for each vertex, the neighborhood of this radius has to contain at least a minimum number of vertices.

Formally, we can use the following probabilistic distance metric to evaluate whether two vertices are in each other’s ε neighborhood:

$$Pr\{J_dis(\sigma_1, \sigma_2) \leq \varepsilon\} \geq \beta \quad (5)$$

where $J_dis(\sigma_1, \sigma_2) = 1 - J(\sigma_1, \sigma_2)$ is *Jaccard distance* function on possible worlds. When we specify the *Jaccard distance* threshold ε as $1 - \tau$ and the probability threshold β as α , where τ and α is the parameters of Eq.(1), we can easily conclude that Eq.(5) holds if and only if Eq.(1) holds. Therefore, when two vertices of G are connected, they are in each other’s $\varepsilon = 1 - \tau$ neighborhood with probability above $\beta = \alpha$. Based on the above observations, we present our density-based clustering algorithm *pro-ER* in Algorithm 2.

The algorithm takes G and the core vertex constraint μ as input. It starts with an arbitrary starting point p that has not been visited. Then it mark p as “visited” (Line 3) before checking its connectivity degree. If the connectivity degree of p is

Algorithm 2. The *pro-ER* algorithm

```

Input:  $G = (V, E)$  and core vertex constraint threshold  $\mu$ 
Output: A participation of vertices
1  $visited[|G|] \leftarrow 0$ ,  $C \leftarrow \emptyset$ ;
2 for each  $p \in V$  such that  $visited[p] = 0$ 
3    $visited[p] = 1$ ;
4   if  $|AdjVex(p)| \geq \mu$ 
5      $C \leftarrow C \cup \{p\}$ ;           //create a new cluster  $C$  add  $p$  to it
6      $N \leftarrow N \cup \{AdjVex(p)\}$ ; //add all connected vertices of  $p$  to  $N$ 
7     for each vertex  $p' \in N$ 
8       if  $visited[p'] = 0$ 
9          $visited[p'] = 1$ ;
10        if  $|AdjVex(p')| \geq \mu$ 
11           $N \leftarrow N \cup \{AdjVex(p')\}$ ;
12          if  $visited[p'] \neq -1$     //have not been in any cluster
13             $C \leftarrow C \cup \{p'\}$ ;
14        end for
15        output  $C$ ;
16        for each vertex  $p' \in C$ 
17           $visited[p'] = -1$ ;
18      else
19        output  $p$  as noise,  $visited[p'] = -1$ ;
20 end for

```

larger than μ (Line 4), this vertex is a core vertex. A new cluster C containing p (Line 5) and a candidate set N containing p 's μ neighborhood vertices is created (Line 6). Then all the unprocessed vertices v in N are tested. For the testing of v , if it has not been visited, *pro-ER* marks it as “visited” and its neighborhood are checked. If v is a core vertex, v 's neighborhoods are added to N (Lines 10-11). The algorithm adds objects to vertices to C until C can no longer be expanded. It means that N is empty. At this time, cluster C is generated and outputted. Then the algorithm continues with a new start vertex to generate a new cluster. Note that, when a vertex is included in a cluster C or recognized as a noise vertex, we mark it to ignore it in following steps (Lines 17 and 19). The time complexity of the algorithm is $O(n + e)$, where e is the number of edges and n is the number vertices.

5 Experimental Study

In this section, we evaluated the efficiency and effectiveness of our algorithms. First, we examined the effectiveness of *wpro-PJOIN* in reducing the number of verification. Second, we tested the quality of *pro-ER* in terms of both precision and recall rate. Finally, the scalability of *pro-ER* was presented.

Experimental Setup. Restaurant¹ is a data set consisting of 864 non-identical restaurant records among which 112 pairs refer to the same entity. Each restaurant record has four attributes [name, address, city, type]. We extended Restaurant as follows.

First, for a deterministic record, by replacing some minor mistake tokens, we can extend this record to a subset of records referring to the same entity. Then, we can get a new dataset *ext-Restaurant* composed of various subsets of deterministic records referring to the same entity. Then we extend *ext-Restaurant* to get various uncertain datasets by randomly replacing grams of each deterministic record.

All the experiments are conducted on a PC with Intel(R) Core(TM) i5-2400 @3.10GHz CPU with 4G memory and a Linux operation system.

5.1 Pairwise Entity Resolution Evaluation

In this subsection, we present performance of *wpro-PJOIN* in terms of pruning ratio of possible worlds on an uncertain dataset *gau-Restaurant* following *Gaussian distribution*. We define the pruning ratio as $(pw - vpw)/pw$ where pw is size of possible worlds and vpw is number of verified possible worlds. The following experiments concern this ratio w.r.t different similarity and probability thresholds.

Performance vs. Similarity Threshold τ . Figure 1 illustrates pruning ratio w.r.t different τ and fixed $\alpha = 0.6$. We present both average ratio of all “similar” pairs and average ratio of all “dissimilar” pairs. An observation is that the join algorithm’s “dissimilar” pruning ratio increases with increasing τ , while the “similar” pruning ration varies inversely. Also the pruning ratio of “similar” pairs differs with that of “dissimilar” pairs significantly. Note that, the join algorithms can achieve high pruning ratio.

Performance vs. Probability Threshold α . Figure 2 illustrates performance w.r.t different α and fixed $\tau = 0.5$. We can see that the algorithm’s “dissimilar” pruning ratio increases monotonically with increasing α , while their “similar” pruning ration decreases almost linearly. Like Figure 1, the “similar” ratio also differs significantly with “dissimilar” ratio. Besides, the average ratio for all pairs is also very high.

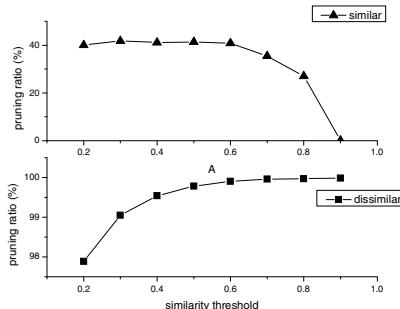


Fig. 1. Pruning ratio vs. τ

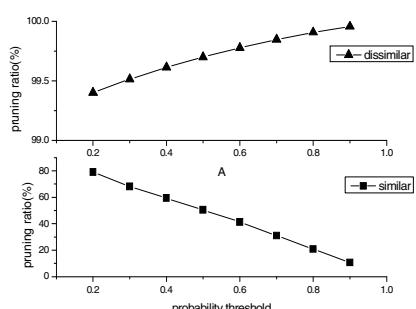


Fig. 2. Pruning ratio vs. α

¹ <http://www.cs.utexas.edu/users/ml/riddle/data/restaurant.tar.gz>

5.2 Group-Wise Entity Resolution Evaluation

In this section, in terms of clustering quality and scalability, we present experimental results on our group-wise entity resolution algorithm *pro-ER*. The algorithm has three parameters (τ, α, μ) . We first present its quality w.r.t various combinations of these parameters and then scalability of the algorithm is explored.

Clustering Quality vs. Different (τ, μ) . Table 1 (“cons” is core vertex constraint) illustrates quality of *t pro-ER* over a Gaussian dataset with fixed $\alpha = 0.6$. Firstly, we can see that with increasing similarity threshold (simi), the precision (pre) increases while the recall (rec) decreasing. Second, with proper (μ, τ) , i.e., (1,0.3), our algorithm can achieve both high precision and recall in the same time. This demonstrates the excellent performance of the algorithm. It is because that in pairwise joins we computed grams’ weight by the *inverse document frequency* weight scheme (IDF) [6].

Table 1. Clustering quality with wpro-PJOIN

simi cons \	0.1		0.2		0.3		0.4		0.5		0.6	
	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec	Pre	Rec
1	0.26%	100.0%	35.42%	97.68%	94.18%	90.48%	99.18%	78.81%	99.63%	63.75%	100.0%	53.99%
2	0.26%	99.64%	35.33%	96.37%	94.65%	88.39%	99.22%	75.30%	99.60%	59.64%	100.0%	48.75%
3	0.27%	99.23%	34.34%	91.37%	94.81%	79.40%	99.04%	61.19%	99.39%	38.75%	100.0%	23.69%
4	0.28%	97.86%	32.59%	83.75%	93.96%	67.62%	99.28%	49.05%	100.0%	24.17%	100.0%	9.76%
5	0.28%	97.62%	31.34%	73.81%	97.51%	53.69%	100.0%	40.71%	100.0%	19.29%	100.0%	8.04%
6	0.28%	96.01%	25.61%	52.86%	97.84%	29.70%	100.0%	23.75%	100.0%	11.25%	100.0%	6.25%

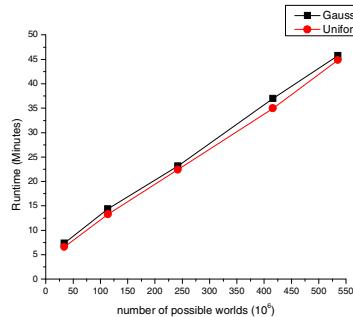


Fig. 3. Scalability results

Scalability vs. Possible World Size. To probe the scalability of *pro-ER*, we tested its runtime over 10 uncertain datasets (5 following *Gaussian distribution* and another 5 *Uniform distribution*). The sizes of possible worlds of all uncertain strings for *Gaussian distribution* datasets are 34M, 114M, 242M, 416M and 535M respectively, similar to the *Uniform distribution* datasets. Figure 3 illustrates runtime of the algorithm w.r.t different possible world size and probability distribution. The parameters of the algorithm is set as $\tau = 0.5$, $\alpha = 0.6$ and $\mu = 3$. Clearly, experiment shows that runtime of the algorithm exhibits a linear growth.

Acknowledgements. This paper was partially supported by NGFR 973 grant 2012CB316200, NSFC grant 61003046, 60933001 and NGFR 863 grant 2012AA011004, IBM CRL Joint Project MH20110819.

References

1. Arasu, A., Ganti, V., Kaushik, R.: Efficient exact set-similarity joins. In: VLDB 2006 (2006)
2. Bayardo, R.J., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: WWW 2007 (2007)
3. Chandel, A., Hassanzadeh, O., Koudas, N., Sadoghi, M., Srivastava, D.: Benchmarking larative approximate selection predicates. In: SIGMOD 2007 (2007)
4. Chaudhuri, S., Ganti, V., Kaushik, R.: A primitive operator for similarity joins in data cleaning. In: ICDE 2006 (2006)
5. Li, C., Lu, J., Lu, Y.: Efficient merging and filtering algorithms for approximate string searches. In: ICDE 2008 (2008)
6. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. Journal of the American Society for Information Science 27(3), 129–146 (1976)
7. Sarawagi, S., Kirpal, A.: Efficient set joins on similarity predicates. In: SIGMOD 2004 (2004)
8. Xiao, C., Wang, W., Lin, X., Shang, H.: Top-k Set similarity Joins. In: ICDE 2009 (2009)
9. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW 2008 (2008)

Imputation for Categorical Attributes with Probabilistic Reasoning

Lian Jin, Hongzhi Wang, and Hong Gao

Department of Computer Science and Technology

Harbin Institute of Technology

msn19882009@live.cn, {wangzh, honggao}@hit.edu.cn

Abstract. Since incompleteness affects the data usage, missing values in database should be estimated to make data mining and analysis more accurate. In addition to ignoring or setting to default values, many imputation methods have been proposed, but all of them have their limitations. This paper proposes a probabilistic method to estimate missing values. We construct a Bayesian network in a novel way to identify the dependencies in a dataset, then use the Bayesian reasoning process to find the most probable substitution for each missing value. The benefits of this method include (1) irrelevant attributes can be ignored during estimation; (2) network is built with no target attribute, which means all attributes are handled in one model;(3) probability information can be obtained to measure the accuracy of the imputation. Experimental results show that our construction algorithm is effective and the quality of filled values outperforms the mode imputation method and kNN method. We also verify the effectiveness of the probabilities given by our method experimentally.

Keywords: missing value imputation, probabilistic reasoning, Bayesian Network.

1 Introduction

In real-life databases, missing value is a common problem due to many unpredictable circumstances such as human carelessness, failing to import data because of violating rules or out of valid rage, or some external factors limiting the collection of data. These missing values bring difficulties into data mining and analysis, which are usually unable to deal with missing values directly.

Instead of discarding all missing data, imputation is a more desirable way to handle datasets before analysis. The simplest way to fill in missing values is to replace them with the mean or the most frequent value. In spite of its simplicity, it works well in many circumstances. In addition to this naïve method, more advanced methods have been proposed to compute more accurate values, which can be divided into two categories.

Statistical Method: The missing value problem has been extensively studied in the statistical area. Many existing models can be applied, such as expectation-maximization

algorithm[1], regression-based imputation[2,3] and sampling method[4]. A more superior method—multiple imputation(MI), which taking uncertainty of each imputation into consideration to avoid bias was first proposed by Rubin[5]. The main problem of this method is its high computation cost.

Machine Learning Method: The nature of learning from existing information empowers many models in this area to solve the missing value problem. Classification methods[9-11], clustering methods[6,7] and neural network methods[12] are all widely researched. The common procedure of these methods is training model from complete data subset and then predicting missing values based on that model.

Even though many methods have been proposed, this problem is not solved throughout. Most of the proposed methods are only suitable for numeric data and it is difficult to simply extend them to the categorical data, which is also an important part of missing value problem. Thus we focus on categorical attributes in this paper.

Among the existing techniques for category attributes, classification is the most commonly used technique [9-11]. Such methods have following drawbacks: (1) Generally, one classification process is targeting on only one attribute(Class), which means only one attribute's value could be estimated. This is inefficient when we want all missing values from different attributes to be filled. (2) It is difficult to deal with high dimensional data. Involving irrelevant attributes may cause unnecessary computation cost and affect the classification quality as well. (3) Without true values for comparison, it is hard to measure the quality or accuracy of the imputed results.

This paper aims to solve the above problems. We propose a novel Bayesian network construction algorithm different from k2 algorithm[17], then use the probabilistic reasoning to estimate missing values. In our method, to solve problem (1), we do not fix any target attribute during the construction of BN. All nodes are added to the net according to their relativity with the others and the most influencing one comes to the root. About problem (2), only the information from the neighbors is used when reasoning for one attribute. The conditional independence helps to cut off all the irrelevant attributes. To solve problem (3), we assign a probability according to the result of reasoning process for every imputation. From experimental results, the higher the probability, the more accurate the imputation will be. It means that we can evaluate the imputation quality from its probability information.

The rest of this paper is organized as follows: Section 2 reviews some basic knowledge about BN and probabilistic reasoning. Section 3 proposes our imputation method. Experiments and analysis are shown in Section 4. Section 5 draws the conclusions.

2 Bayesian Network and Probabilistic Reasoning

Before presenting our algorithm in detail, here we introduce some basic knowledge about Bayesian network and probabilistic reasoning, which are used in our algorithm.

2.1 Bayesian Network

A Bayesian Network (BN for brief) is a directed acyclic graph whose nodes represent variables and edges represent causal relationships. The causality is given by the conditional probability $P(x_i|\pi(x_i))$, in which x_i is the i th variable and $\pi(x_i)$ is its parents. In the assumption of conditional independence, the joint probability distribution of x_1, x_2, \dots, x_n can be computed by multiplying all their conditional probabilities:

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i|\pi(x_i)) \quad (1)$$

When $\pi(x_i) = \emptyset$, $P(x_i|\pi(x_i)) = P(x_i)$. Such conditional independence assumption reduces the model complexity greatly and makes it easier for reasoning process, especially when dealing with high dimensional data.

2.2 Probabilistic Reasoning

Probabilistic reasoning is to infer the unknown information from given information using probabilistic knowledge. There are two major problems on BN for reasoning: (1) posterior probability reasoning, which infers the posterior probability distribution of some variables and (2) maximum a posteriori hypothesis(MAP) problem, which finds the most possible value combination of some variables. This paper handles the latter problem, whose formal definition is given in Eq (2)

$$h^* = \operatorname{argmax}_h P(H = h|E = e) \quad (2)$$

Eq (2) means to find the most possible compound status(H) under the condition of $E=e$. More about probabilistic reasoning are discussed in [16].

3 Missing Value Imputation

In this section, we discuss our missing value imputation algorithm. Our imputation algorithm consists of two phases, finding dependencies and estimating missing values. We build a BN in the first phase. Dependency relationships and relevant parameters will be computed in this phase, as discussed in Section 3.1. Next, a probabilistic reasoning is used to complete the estimation part. Section 3.2 shows the details.

3.1 Construction of Bayesian Network

Before the reasoning process, a BN needs to be constructed. Many algorithms have been proposed to learn a BN automatically from data, such as k2 and hill climb. But as we have examined, for many datasets, these methods generate too many independent nodes which is not suitable for efficient reasoning.

As an example for this point, Table 1 gives the BN constructed from k2 on one of our experimental dataset from UCI. For simplicity, we only use part of this dataset. It is observed that without any target attribute, all nodes are isolated, which disables the reasoning process. This is because the score function used in k2 algorithm is too strict

to connect some nodes with correlativity. To solve this problem, we change the score function. Since our goal is not to build a real causal net but to find relevant attributes for imputation, we try to explore as strong dependencies as possible, so that the missing nodes can get as much information as possible from the most relevant neighbors. Here we use the τ -coefficient as the score function which is a correlation coefficient between two variables proposed by Goodman and Kruskal[15]. It represents the reduction of the prediction error on Y when X is known. Eq (3) gives the definition of τ -coefficient according to Table 2.

$$\tau = \frac{\sum_{i=1}^c \sum_{j=1}^r \frac{n_{ij}^2}{n_{i*}} - \frac{1}{n} \sum_{j=1}^r n_{*j}^2}{\sqrt{\frac{1}{n} \sum_{j=1}^r n_{*j}^2}} \quad (3)$$

n_{*j} , n_{i*} and n in Table 2 represents $\sum_{i=1}^c n_{ij}$, $\sum_{j=1}^r n_{ij}$ and $\sum_{i=1}^c \sum_{j=1}^r n_{ij}$ respectively. Algorithm1 shows the main procedure of constructing a BN.

Table 1. Bayesian Networks Constructed on “mushroom”

K2			Our algorithm		
NO	name	parents	NO	name	parents
0	class	-1	0	class	3
1	cap-color	-1	1	cap-color	5,4,0
2	bruises	-1	2	bruises	3
3	stalk-surface-above-ring	-1	3	stalk-surface-above-ring	4
4	stalk-surface-below-ring	-1	4	stalk-surface-below-ring	5
5	stalk-color-below-ring	-1	5	stalk-color-below-ring	-1
6	ring-number	-1	6	ring-number	5,1,3

Table 2. Contingency Table of X and Y

	x_1	x_2	...	x_c	n_{*j}
y_1	n_{11}	n_{21}	...	n_{c1}	n_{*1}
y_2	n_{12}	n_{22}		n_{c2}	n_{*2}
...					
y_r	n_{1r}	n_{2r}	...	n_{cr}	n_{*r}
n_{i*}	n_{1*}	n_{2*}	...	n_{c*}	n

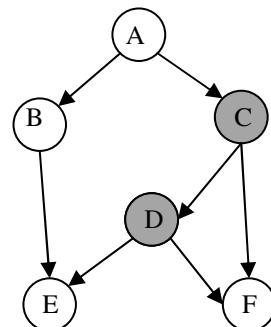


Fig. 1. Bayesian Networks

Algorithm1:BNConstruction

Input: dataset $D=\{A_1, A_2, \dots, A_n\}$ A_i is the i th attribute

Output: Bayesian network

1. **for** each pair of A_i and A_j ($i \neq j$)

2. $t_{ij} = \tau$ -coefficient of A_i and A_j

```

3.  $A_1', A_2', \dots, A_n'$ =sort  $A_1, A_2, \dots, A_n$  according to  $t_{i*}$  ( $t_{i*} = \sum_{j=1}^N t_{ij}$ ,  

   where  $N$  is the data size) in descending order.
4. for  $i=1$  to  $n$ 
5.   create node for  $A_i'$ 
6.    $old-score=0$ 
7.    $parent=\{\}$ 
8.   while true
9.     for  $j=1$  to  $i-1$ 
10.       $add-node=\{\}$ 
11.      new-score= $\tau$ -coefficient of  $A_i'$  and  $parent \cup A_j'$ 
12.      if new-score> $old-score$ 
13.         $old-score=new-score$ 
14.         $add-node=add-node \cup A_j'$ 
15.      if  $add-node \neq \{\}$ 
16.        add  $add-node$  to  $parent$ 
17.      else
18.        break
19.   add edges from  $parent$  to  $A_i'$ 
20.   computes  $p(A_i' | parent)$ 

```

In Algorithm 1, line 1-3 give an order of attributes for building the net according to their coefficient matrix. Line 4-18 find parents for each node that can help to improve its classification power. Line 20 computes the corresponding conditional probability.

Table 3. Coefficient Matrix

NO	0	1	2	3	4	5	6	sum	order
0	—	0.007	0.252	0.254	0.205	0.049	0.036	0.803	4
1	0.048	—	0.047	0.107	0.119	0.115	0.135	0.571	6
2	0.252	0.003	—	0.266	0.219	0.033	0.001	0.774	5
3	0.346	0.033	0.331	—	0.335	0.112	0.021	1.178	3
4	0.33	0.043	0.321	0.367	—	0.111	0.015	1.187	2
5	0.265	0.094	0.243	0.304	0.278	—	0.237	1.421	1
6	0.046	0.011	0.005	0.011	0.015	0.029	—	0.117	7

We use the data in Table 1 to demonstrate our algorithm. Table 3 is the relativity coefficient matrix computed in line 1-2. The last column shows the ordered numbers after line 3. According to the algorithm, since there is no attribute ordered before 5, it is checked firstly with doing nothing. The iteration moves to the next attribute 4. Line 9-18 go through {5} for finding 4's *parent*. We get $\tau(5,4) = 0.278 > 0 = old-score$, so 5 is added to 4's *parent*. When 3 is to be processed, the most relevant attribute from {5,4} is picked firstly by line 9-18. Since $\tau(4,3) > \tau(5,3) > 0$, we add 4 to 3's *parent* and set *old-score* to 0.367. In the second loop, line 9-14 try each node

left in $\{5\}$, combining it with the current *parent* set and getting $\tau(\{4,5\},3) = 0.272 < old-score$, which stops the iteration. The similar procedure goes for the rest nodes. In the algorithm, we set an upper bound for the size of *parent* set, to avoid unnecessary computation caused by too many edges. The final result is shown in Table 1, from which it can be observed that each node except for the root has its parents, which means more relationships are explored compared to k2.

Complexity Analysis. Each computation of the coefficient needs one pass scan of the dataset. The time complexity is $O(n)$, where n is the size of dataset. Therefore, line 1 and line 2 cost $O(m^2n)$, where m is the number of attributes. Testing each parent for one node requires $O(n)$ and the i th node has $(i-1)$ probable parents which means the time complexity of line 4-18 is $O(m^2n)$. The run time of line 20 is $O(n)$. To sum up, the total complexity of Algorithm1 is $O(m^2n)$.

3.2 Missing Value Imputation Using Probabilistic Reasoning

After all dependencies are computed as BN, the next step is to estimate each missing value through probabilistic reasoning. The main idea is to find the most possible substitution for a missing value with given evidences. For each missing value (H) of each record, the first step is to find all relevant attributes (O), whose values are known. Then the value of H that maximize $P(H|O)$ is computed. Equation (4-8) explains the main derivation of our imputation method.

Assume U is the attribute set of a dataset D . For each instance I , we define a missing set $H = \{h|h \in U \text{ and } I_h = "?"\}$ and a observed set $O = \{o|o \in U \text{ and } I_o \neq "?"\}$, where I_x means the value I is assigned to attribute x and “?” means missing value. Our task is to complete each h_i in H given O . according to Eq (2), our goal is to compute

$$\operatorname{argmax}_{h_i} P(h_i|O) \quad (4)$$

With Eq (4), each h in H could be computed separately. Through Bayesian transformation, we have $P(h_i|O) = \frac{P(h_i,O)}{P(O)}$. Ignoring the constant term $P(O)$, Eq (4) equals to solve:

$$\operatorname{argmax}_{h_i} P(h_i, O) \quad (5)$$

$$= \operatorname{argmax}_{h_i} \sum_{h_1} \dots \sum_{h_{i-1}} \sum_{h_{i+1}} \dots \sum_{h_m} P(H, O) \quad (6)$$

Since $H \cup O = U$, we have $P(H, O) = P(U)$. We split this probability according to Eq (1):

$$\operatorname{argmax}_{h_i} \sum_{h_1} \dots \sum_{h_{i-1}} \sum_{h_{i+1}} \dots \sum_{h_m} \prod_{u \in U} P(u|\pi(u)) \quad (7)$$

After eliminating the constant term and integrate all h in $H-h_i$, we get

$$\operatorname{argmax}_{h_i} \{P(h_i|\pi^{obv}(h_i)) * \prod_{\substack{ch \in \text{child}(h_i) \\ \text{and } ch \in H}} P(ch \cup \pi^{obv}(ch)) * \prod_{\substack{ch \in \text{child}(h_i) \\ \text{and } ch \in O}} P(ch|h_i \cup \pi^{obv}(ch))\} \quad (8)$$

The above transformation helps to reduce the computation complexity greatly. It only reserves the observed parent nodes, child nodes and their parent nodes which form the Markov boundary when integrating the unknown ones. For every instance, we estimate each missing value separately, which means we abandon the relationship between unknown variables ($h_i \rightarrow h_j$). It cuts off the uncertainty transmitted from unknown “neighbors”, which helps to improve the imputation accuracy. The pseudo code of probabilistic reasoning is shown in Algorithm2.

After each reasoning, we assign a probability to the imputation value. The probability obtained from reasoning process cannot be used directly, because these probabilities may have great difference due to the number of conditional probabilities involved in Eq (8). We choose the relative probability here:

$$p(a_i) = \frac{\max_{v_k} p(a_i=v_k)}{\sum_{v_k} p(a_i=v_k)} \quad (9)$$

Where $p(a_i = v_k)$ is the probability obtained from reasoning process. Eq(9) describes the certainty of the imputed value comparing to the rest candidates.

Algorithm2 :Probabilistic reasoning

Input: Dataset D with missing values
Output: completed dataset D'

1. **for** each r in D
2. $missSet = \{i|r_i = "?"\}$
3. **for** each i in $missSet$
4. $pSet = \{\}$
5. $pSet = pSet \cup p(i|Parent_i^{obv})$
6. **for** each i 's child-node ch
7. **if** ch is missing
8. $pSet = pSet \cup p(i \cup Parent_{ch}^{obv})$
9. **else**
10. $pSet = pSet \cup p(ch|i \cup Parent_{ch}^{obv})$
11. $r_i = argmax_{v_i} \prod_{p \in pSet} p$

Line 5-10 are the major part, which collect the related probabilities for each node. Line 5 handles the parent nodes and line 6-10 deal with the child nodes.

For explanation, we take the BN shown in Fig 1 as an example. For some instance I , its values on variable C and D are lost. We estimate C firstly, under the knowledge of A, B, E and F . Its $pSet$ is initialized to empty in line 4. Line 5 forms a conditional probability from C 's observed parent node, whose result is $P(C|A)$. Line 6-10 handle C 's child nodes $\{D, F\}$ one by one. Since D is missed, line 8 is executed, which adds $P(C)$ to the $pSet$. When checking for F , line 10 is executed and gets another conditional probability $P(F|C)$. After this, all parameters related to C have been gathered, so we come to line 11 to find the most possible substitution for C that maximize $P(C|A) * P(C) * P(F|C)$, where the value of A and F are known. Attribute D goes through the same procedure, which gets $pSet=\{P(D), P(E|D), P(F|D)\}$ from line 5-10 and solves $argmax_d P(D) * P(E|D) * P(F|D)$ in line 11.

Time Complexity Analysis. Since this algorithm processes missing values one by one, its time complexity is $O(|D| * m * d * |pSet|)$, where m is the number of missing value in one record and d is the domain size for one attribute. Since compared with $|D|$, the effect of m , d , and $|pSet|$ can be treated as constant, the complexity of algorithm 2 can be considered as $O(|D|)$, which means algorithm2 is linear with data size.

4 Experimental Results

In this section, we show the effectiveness of the proposed algorithm on real datasets. All algorithms are implemented with c++. The algorithms are run in a PC with 2GH cpu and 1G main memory. The operation system is WinXP. The basic information of datasets we used in this paper is shown in Table 4. They are from UCI machine learning repository¹. We use categorical attributes in the data set to test our algorithms. All missing values in the original datasets are removed beforehand. Their #variable and size given by Table 4 are those after removing.

Table 4. Datasets used in experiments(from UCI)

Dataset	#Variable	Size
Zoo	18	101
Mushroom	22	8124
Adult	15	30162

4.1 Effectiveness of BN

In this part, we test the effectiveness of our BN. For comparison, we build another BN that places one target attribute at the root and adds the rest attributes by k2 algorithm. We name it as sep-BNs, while ours as sin-BNs. We pick some attributes from each dataset as representatives. For each attribute, we eliminate 10% values and use these two methods to predict them. The results are shown in Fig2.

The experiment proves that comparing with multiple executions of sep-BNs, the classification ability is hardly weakened for most attributes in sin-BNs when all of them are classified in one net at the same time.

As we have discussed in 3.1, in most circumstances, the k2 algorithm tends to form a naïve BN, in which the target node is connected to almost all the other nodes. While in our algorithm, only the information from neighbors is used. As can be seen from the results, their effects are almost equal. The reduction in computation does not sacrifice the final precision too much. It means that it is unnecessary to involve more variables than need. Our method is able to cut down the irrelevant attributes appropriately for each node.

¹ <http://archive.ics.uci.edu/ml/>

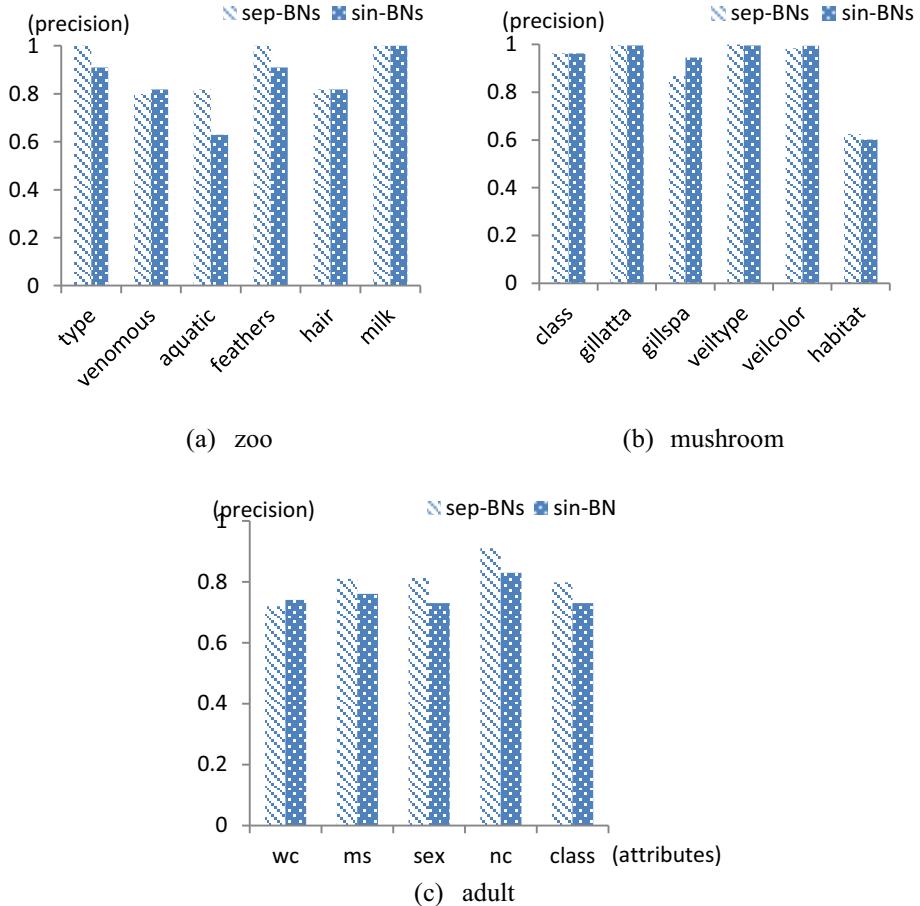


Fig. 2. ClassificationPrecisions of Three Datasets

4.2 Estimation Accuracy

We compare our algorithm with two most commonly used methods—mode imputation and kNN method. In mode imputation, we replace the missing values with the most frequent value. In kNN, we choose candidate with the maximum probability as substitution. The number of clusters is determined by the domain size of each attribute. With this criterion, we clustered “zoo” into 6, “mushroom” into 8 and “adult” into 6. For each dataset, we generate 5 test set. For one of them ,we pick the attributes used in 4.1 and generate missing values with different missing rates. For the rest 4 datasets, we generate missing values among the whole dataset in the assumption that all values are missed at the same probability. They only varied in missing rate. Table 5 gives their imputation precisions.

From Table 5, it can be observed that our method outperforms the mode and kNN method. The precision of mode and sin-BNs change little with missing rates when the data are missed at the same probability, in which condition the distribution of one attribute and the dependencies between attributes are mostly reserved. While in kNN, the more missing values, the fewer information it used for clustering. So it is more sensitive to the missing rate. In the “mix” test set, where we remove the impact of weakly related attributes, the superiority of the sin-BNs is shown out under comparison.

Table 5. Imputation Precision of Each Method on Different Datasets

DATASET	MISRATE%	METHOD		
		mode	KNN	sin-BNs
zoo	5%	74.5	86.2	90.0
	10%	69.1	83.4	87.7
	20%	67.5	74.2	84.0
	30%	61.5	74.0	85.6
	mix	65.4	80.0	88.0
mushroom	5%	59.2	74.5	75.5
	10%	58.6	73.7	73.4
	20%	58.4	72.9	73.0
	30%	57.3	65.0	76.2
	mix	86.5	93.0	94.0
adult	5%	58.7	61.1	70.0
	10%	58.6	59.5	67.6
	20%	58.6	57.2	60.4
	30%	58.5	46.4	66.4
	mix	54.9	63.4	79.0

4.3 Effectiveness of Probabilistic Reasoning

In this part, we are going to show the effectiveness of the probability given in Eq(9) from two aspects. On one hand, we attempt to verify that the true value always has a relatively high probability and in most cases, it gets the maximum probability. On the other hand, we are going to show that an attribute’s average probability can represent its imputation accuracy.

For the first experiment, we use the “mix” test set in Table 5. We measure the average probabilities of truth-values(p_t) and the percentage of truth-values that has the maximum probability among all candidates. The results are shown in Table 6. The imputation might fail in some conditions due to the lack of knowledge. As can be observed from the result, the reasoning process helps to discover the truth value.

In the second experiment, we eliminate 10% values of “mushroom” to compare its imputation accuracy and average imputed probability. Table 7 shows the results. In Table 7, the average probability almost follows the imputation accuracy, which means the higher the probability, the higher the accuracy. In this sense, we can infer the imputation quality from its probability.

From the above two experiments, we can see that the probability given by our algorithm could be used to represent the validity of an imputation.

Table 6. Results of Experiment4.3

mis-rate%	zoo			mushroom			adult		
	\bar{p}_i/\bar{p}_f	$t_{max}\%$	#failed	\bar{p}_i/\bar{p}_f	$t_{max}\%$	#failed	\bar{p}_i/\bar{p}_f	$t_{max}\%$	#failed
5	0.85/0.06	92	0	0.91/0.04	92	0	0.74/0.08	80	0
10	0.88/0.105	90	3	0.90/0.05	91.2	0	0.74/0.08	78	0
20	0.87/0.11	87	28	0.91/0.04	92	0	0.72/0.08	77	0
mixed	0.86/0.09	88	1	0.93/0.04	94	0	0.75/0.07	79	0

Table 7. ImputationAccuracy and Average Probability of each Attribute in Mushroom

name	accuracy	average probability
class	0.94	0.938
cap-surface	0.486	0.546
bruises	0.845	0.84
stalk-surface-below-ring	0.69	0.73
ring-type	0.99	0.996
habitat	0.67	0.75

5 Conclusion

In this paper, we study the imputation method using probabilistic reasoning. It is processed on a BN, which is constructed in a different way to explore more correlations among all attributes in our algorithm. Experimental results show that (1)our BN performs well on classification problem,(2) the missing rate affects little on our imputation algorithm when the missing values are randomly distributed and(3) the probability information can reflect the imputation accuracy. Our future work includes taking mixture attributes into consideration and performing more experiments and comparisons.

Acknowledge. This paper was partially supported by NGFR 973 grant 2012CB316200, NSFC grant 61003046 and NGFR 863 grant 2012AA011004. Doctoral Fund of Ministry of Education of China (No.20102302120054). the Fundamental Research Funds for the Central Universities(No. HIT. NSRIF. 2013064).

References

1. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B* 39, 1–39 (1977)
2. Yang, K., Li, J., Wang, C.: Missing Values Estimation in Microarray Data with Partial Least Squares Regression. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) *ICCS 2006. LNCS*, vol. 3992, pp. 662–669. Springer, Heidelberg (2006)
3. Shan, Y., Deng, G.: Kernel PCA regression for missing data estimation in DNA microarray analysis. In: *IEEE International Symposium on Circuits and Systems, ISCAS 2009*, May 24-27, pp. 1477–1480 (2009)
4. Walsh, B.: Markov chain Monte Carlo and Gibbs sampling. Lecture notes for EEB 581, version 26 (April 2004)
5. Little, R., Rubin, D.B.: *Statistical Analysis With Missing Data*. Wiley, New York (1987)
6. Zhang, S.: Shell-neighbor method and its application in missing data imputation. *Applied Intelligence* 35(1), 123–133 (2011)
7. Ling, W., Mei, F.D.: Estimation of missing values using a weighted k-nearest neighbors algorithm. In: *Proceedings - 2009 International Conference on Environmental Science and Information Application Technology, ESIAT 2009*, vol. 3, pp. 660–663 (2009)
8. Yuan, Y.C.: Multiple Imputation for Missing Data: Concepts and New Development (Version 9.0). SAS Institute Inc., NC (2001), <http://www.sas.com/statistics>
9. Lakshminarayanan, K., Harp, S.A., Goldman, R., Samad, T.: Imputation of missing data using machine learning techniques. In: Simoudis, Han, Fayyad (eds.) *Proceedings: Second International Conference on Knowledge Discovery and Data Mining*, pp. 140–145. AAAI Press, Menlo Park (1996)
10. Zhang, S., Qin, Z., Ling, C.X., Sheng, S.: “Missing is useful”: missing values in cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering* 17(12), 1689–1693 (2005)
11. Li, X.-B.: A Bayesian approach for estimating and replacing missing categorical data. *Journal of Data and Information Quality* 1(1) (2009)
12. Setiawan, N.A., Venkatachalam, P., Hani, A.F.M.: Missing Attribute Value Prediction Based on Artificial Neural Network and Rough Set Theory. In: *International Conference on BioMedical Engineering and Informatics, BMEI 2008*, May 27-30, vol. 1, pp. 306–310 (2008)
13. Hruschka, E.R., Hruschka, E.R., Ebecken, N.F.: Bayesian networks for imputation in classification problems. *Journal of Intelligent Information Systems* 29(13), 231–252 (2007)
14. Di Zio, M., Scanu, M., Coppola, L., Luzi, O., Ponti, A.: Bayesian networks for imputation. *Journal of the Royal Statistical Society A* 167(pt. 2), 309–322 (2004)
15. Goodman, L.A., Kruskal, W.H.: Measures of association for cross-classification. *Journal of the American Statistical Association* 49, 732–764 (1954)
16. Pearl, J.: *Probabilistic Reasoning in Intelligent Systems*. In: *Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco (1988)
17. Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9, 309–347 (1992)

A New Approach to Identify Influential Spreaders in Complex Networks^{*}

Qingcheng Hu^{**}, Yang Gao, Pengfei Ma, Yanshen Yin,
Yong Zhang, and Chunxiao Xing

Research Institute of Information Technology, Tsinghua National Laboratory for Information
Science and Technology, Department of Computer Science and Technology,
Tsinghua University, Beijing, China
{hqc10,yang-gao10,mpf11,yys12}@mails.tsinghua.edu.cn,
{zhangyong05,xingcx}@tsinghua.edu.cn

Abstract. In the research of the propagation model of complex network, it is of theoretical and practical significance to detect the most influential spreaders. Global metrics such as degree centrality, closeness centrality, betweenness centrality and K-shell centrality can be used to identify the influential spreaders. These approaches are simple but have low accuracy. We propose K-shell and Community centrality (KSC) model. This model considers not only the internal properties of nodes but also the external properties of nodes, such as the community which these nodes belong to. The Susceptible-Infected-Recovered (SIR) model is used to evaluate the performance of KSC model. The experiment result shows that our method is better to identify the most influential nodes. This paper comes up with a new idea and method for the study in this field.

Keywords: Complex networks, Centrality measures, Influential node, SIR model.

1 Introduction

Many systems perform like complex networks, such as the Internet, social net-work, computing networks, biological network and social system. There are many re-searches on the topology and functionality of the complex networks. It is valuable to identify the most influential nodes [1-2]. This will help to control the disease transmission, rumors spreading [3], computer virus spreading and popularize new ideas and new products.

In this paper, we propose the KSC (K-shell and Community centrality) model. This model considers not only the internal properties of nodes but also the external properties of nodes. The internal properties mean the classic centralities such as degree, closeness, betweenness and so forth. The external properties mean the properties based on community, such as the size and closeness of the community which the node

^{*} Supported by National Basic Research Program of China (973 Program) No.2011CB302302;
Tsinghua University Initiative Scientific Research Program.

^{**} Corresponding author.

is belong to. Then we use the Susceptible-Infected-Recovered (SIR) model to evaluate the performance of our model. The experiment results shows that our method is much better to identify the most influential nodes.

2 Related Work

Identifying influential spreaders has remarkable practical value in complex networks. It can help people control the disease transmission, rumors spreading, computer virus spreading and popularize new ideas and new products.

2.1 K-Shell Decomposition Method

K-shell decomposition [4] is a well-established method for analyzing the structure of large-scale graphs, denoted by $C_{ks}(v)$. A step of decomposition is performed by repeatedly deleting all nodes with degree less than k . The K-shell value of the nodes removed in this step is k . The whole decomposition is finished when all nodes are removed. K increases from 1.

2.2 Community Detecting Algorithms

Network community structure [5] is one of the most fundamental and important topological properties of complex networks, within which information spreading is faster than between which they are quite sparse. In this paper, we implement FN algorithm [6] to detect communities.

3 KSC Centrality Model

We think that a node's influence is determined not only by the node's internal properties but also by the node's external properties. This is consistent with the philosophical internal cause and external cause. KSC is a novel idea which combines the internal properties with the external properties.

Given a complex network $G=(V, E)$, the KSC value of node v_o is denoted by:

$$KSC(v_o) = \alpha f_{internal}(v_o) + \beta f_{external}(v_o), \alpha+\beta=1 \quad (1)$$

$f_{internal}(v_o)$ represents the node's internal influence while $f_{external}(v_o)$ represents the node's external influence, α is the internal factor while β is the external factor, which satisfies $\alpha+\beta=1$, α and β are determined by the actual topology and functionality of the network.

$f_{internal}(v_o)$ is denoted by:

$$f_{internal}(v_o) = \frac{K(v_o)}{\max_{v \in V}(K(v))} \quad (2)$$

$K(v)$ is the internal property, which can be valued with degree, betweenness, closeness and K -shell. $\max_{v \in V} K(v)$ is the normalized factor.

$f_{\text{external}}(v_o)$ is denoted by:

$$f_{\text{external}}(v_o) = \frac{\sum_{c \in C} d(v_o, c) |c|}{\max_{v \in V} (\sum_{c \in C} d(v, c) |c|)} \quad (3)$$

C is the collection composed by the communities calculated by FN algorithm, $d(v_o, c)$ is the number of v_o 's neighbor in community c , $|c|$ is the size of community c , $\max_{v \in V} (\sum_{c \in C} d(v, c) |c|)$ is the normalized factor. $f_{\text{external}}(v_o)$ increases by the number of neighbors which lies in different communities. It indicates v_o 's ability to propagate messages to various communities which is related to the influence of v_o .

α and β are set to different values according to different networks' topology which won't be discussed here. To simplify the experiment while ensuring high performance, in this paper, the experiments use the following configurations:

$$\alpha = \beta = 0.5, \quad K(v) = C_{ks}(v) \quad (4)$$

4 Experiments

4.1 SIR Model

In social networks, SIR model has been widely used in the research of disease, information and rumors spreading. In order to verify our proposed model, we use SIR model [7] to simulate the propagation process, and compare the result with ours.

The SIR model is dynamic in three senses: susceptible, infectious and recovered. When an individual is in infectious sense, it will infect neighbor individuals in susceptible sense by the probability of β . The infected individuals will recover by the probability of γ .

4.2 Dataset

Considering the different social network types representing the different properties of the network topology, we select four real networks dataset for analyzing. (i) Blogs [8]. (ii) NetScience [9]. (iii) Router [10]. (iv) Email [11]. Our model can also be applied to other types of complex networks.

4.3 Performance of the Experiment

We implement the SIR propagation model to evaluate the actual influence of nodes. Only one node is chosen as the initial propagation node in each simulation. The propagation time t is set to 10. The influence $F(t)$ is denoted by the number of nodes in state I and R. We take the average $F(t)$ of 1000 simulations as the final $F(t)$.

The above $F(t)$ is denoted by the actual influence of a node, which is used as a reference when evaluating KSC and other four classic centralities.

Fig. 1. indicates the relation between centrality and actual influence in different networks. In Blogs, the relations between $F(t)$ and the degree, K-shell and betweenness are very weak. For example, in the figure of betweenness, some nodes with higher betweenness are less influential than some others with lower betweenness. However, the KSC centrality is able to distinguish the most influential nodes. In Email, all centralities achieve high performance, while the closeness and KSC are better. This is determined by the topology of the network. In Netscience, betweenness has the worst performance. Closeness and K-shell has little relation with $F(t)$. KSC performs better than degree. In Router, K-shell and KSC achieve similar results. Other centralities are hard to distinguish the influential nodes.

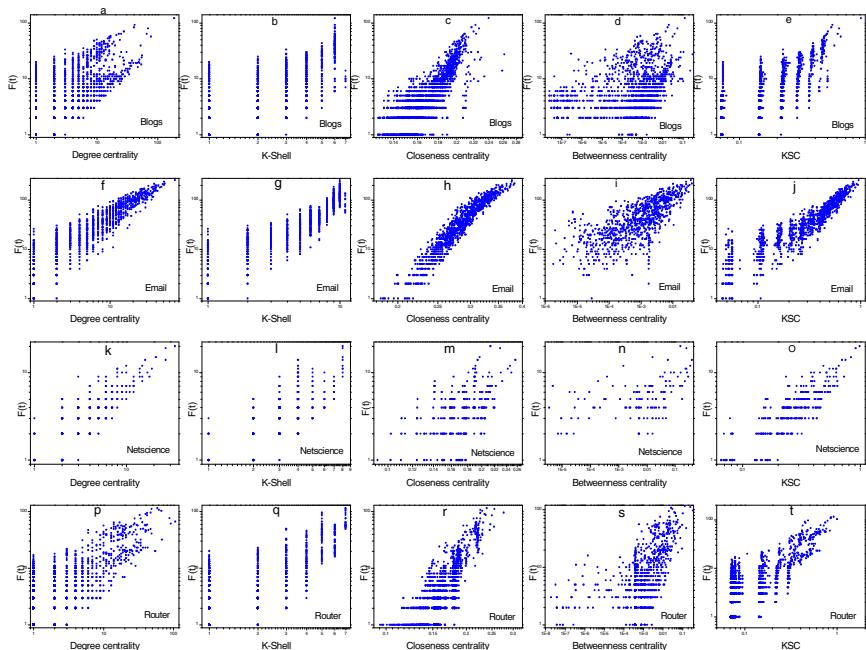


Fig. 1. Comparisons of nodes' influence

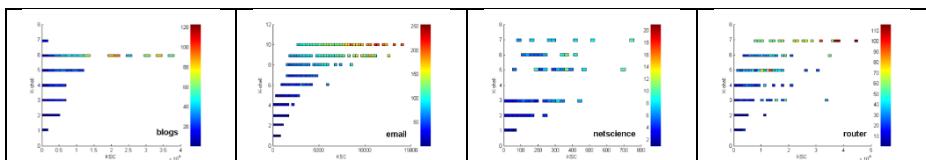


Fig. 2. The comparisons between KSC and K-shell in different complex networks. The horizontal axis represents KSC centrality while the vertical axis represents K-shell centrality. The color at point (x, y) represent the $F(t)$ of the node with $KSC=x$ and $K\text{-shell}=y$.

In one word, the classic centralities have their advantages and disadvantages. KSC is obviously the best in general. KSC is able to distinguish the most influential nodes and is suitable for more complex networks.

[12] points out that in case of single propagation source, the most influential nodes are not the nodes with the highest degree or betweenness, but the nodes with highest K-shell. KSC is based on K-shell, but we take the external properties into consideration.

Fig. 2 shows the comparison between KSC and K-shell. In Email, given K-shell=10, the actual influence $F(t)$ is not constant and has an increasing trend with the increase of KSC. In the other side, when we fix KSC, $F(t)$ is relatively stable, which is indicated by the small range of color change. As we can see from this point, the external properties are important factors to evaluate the most influential nodes.

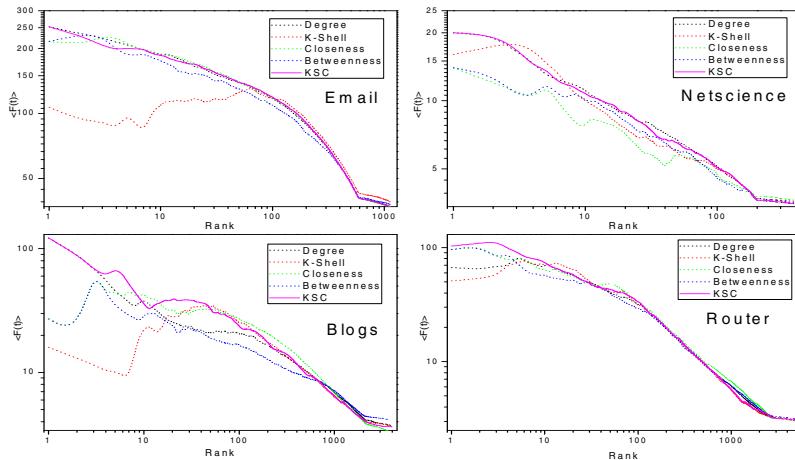


Fig. 3. The average number of $F(t)$ ($t=10$) of the top- k nodes as ranked by the five centrality models. The vertical axis denotes the influence $F(t)$. The horizontal axis denotes the different models rank. The curves of different colors denote different models.

Fig. 3 shows the average number of $F(t)$ ($t=10$) of the top- k nodes as ranked by the five centrality models. For example, in Blogs of Fig. 3, (x, y) is a point (node) on the curve of KSC. The horizontal value of the point represents its influence rank (i.e. x) in KSC model. The vertical value of the point represents its influence (i.e. y) in SIR model. Theoretically, an ideal model gives a monotonically decreasing curve in the figure. If the actual influence $F(t)$ of a node is lower, the influence rank of the node in this model will be lower too.

From the results, we can find that the four classic models fluctuate in different networks, particularly sharply in the top-10 results. In Blogs, the four models are all not accurate. The lower rank nodes are more influential than the higher rank nodes. But the KSC model we proposed almost meets the theoretical curve and the real situation.

The experiments adequately demonstrate that the influence of the nodes is not only determined by internal properties, but also closely related to external properties.

5 Conclusion

Identifying the most influential spreaders in complex networks can help us improve the propagation efficiency of new ideas and products, and develop appropriate strategies to prevent the spread of diseases and rumors. We propose the KSC centrality model. This model is used to analyze the complex networks by considering not only the internal properties of nodes but also the external properties of nodes. We chose four common complex networks, Blogs, Email, Router and Netscience as the datasets of our experiment. In the experiment, we calculate and rank the influence of all nodes in those networks by different models. After that, we use SIR model to simulate the propagation process and compare the result of different models. The experiment results show that our model is more accurate and applicable to identify the most influential spreaders than four classic models.

It is a hot but difficult research to identify the most influential spreaders in complex networks. This paper provides a new idea and method for this challenging work. We hope it can spark the future studies.

References

1. Kempe, D., Kleinberg, J.M., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD 2003, pp. 137–146 (2003)
2. Weng, J., Lim, E.-P., Jiang, J., He, Q.: Twitterrank: finding topic-sensitive influential tweeters. In: WSDM, pp. 261–270 (2010)
3. Budak, C., Agrawal, D., El Abbadi, A.: Limiting the spread of misinformation in social networks. In: WWW, pp. 665–674 (2011)
4. Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., Shir, E.: A model of Internet topology using k-shell decomposition. Proc. Natl. Acad. Sci. USA 104, 11150–11154 (2007)
5. Newman, M.E.J.: Detecting community structure in networks. European Physical Journal (B) 38(2), 321–330 (2004)
6. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Physical Review E 69(6), 066133 (2004)
7. Diekmann, O., Heesterbeek, J.A.P.: Mathematical Epidemiology of Infectious Diseases: Model Building, Analysis and Interpretation. Wiley Series in Mathematical & Computational Biology, New York (2001)
8. Xie, N.: Social network analysis of blogs, M.Sc. Dissertation, University of Bristol (2006)
9. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74, 36104 (2006)
10. Spring, N., Mahajan, R., Wetherall, D., Anderson, T.: Measuring ISP topologies with rocketfuel. ACM SIGCOMM Comp. Comm. Rev. 32(4), 3–145 (2004)
11. Guimerà, R., Danon, L., Díaz-Guilera, A., Giralt, F., Arenas, A.: Self-similar community structure in a network of human interactions. Phys. Rev. E 68, 065103 (2003)
12. Kitsak, M., Gallos, L.K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H.E., Makse, H.A.: Identifying influential spreaders in complex networks. arXiv, Tech. Rep., physics and Society, 1001.5285 (2010)

Presenting XML Schema Mapping with Conjunctive-Disjunctive Views^{*}

Xuhui Li¹, Shafeng Zhu², Mengchi Liu¹, and Ming Zhong¹

¹ State Key Lab of Software Engineering,
Computer School, Wuhan Univ.,
Wuhan, China

{lixuhui,clock}@whu.edu.cn, mengchi@sklse.org

² School of Computer Science and Shanghai Key Lab of Intelligent Information
Processing, Fudan Univ.,
Shanghai, China
zhusf@fudan.edu.cn

Abstract. XML schema mapping is a fundamental issue in XML integration and exchange. Existing studies often follow the approach of relational schema mapping to present XML mapping as correspondences between and constraints on tuples of elements. However, XML mappings involving hierarchical and heterogeneous data are often complex to present with this approach. In this paper, we propose a new approach named Conjunctive-Disjunctive View (CDV) for XML schema mapping. A CDV is a pattern-based view which allows disjunctive as well as conjunctive composition of data elements. It can conveniently present complex mapping requests involving hierarchical, heterogeneous and even recursive data. We describe the design issues and the features of CDV, with certain typical examples of XML schema mappings.

1 Introduction

Schema mapping of XML documents is an interesting and important topic in XML data integration and exchange. In relational data integration[1] and exchange[2], the mapping is often denoted by the source-to-target dependency formula $\varphi_S(\bar{x}) \rightarrow \exists \bar{y}. \psi_T(\bar{x}, \bar{y})$ where φ_S is a conjunction of atoms (an atom is an expression $R(t)$ where R is a relation symbol and t is a tuple of variables) over the source schema S and ψ_T is a conjunction of atoms over the target schema T . XML document is essentially different from relational database in that the data elements in XML document are often in various structural relationships, i.e., the parent-children, the ancestor-descendants, the sibling and the document order relationships. Some studies try to present simple XML schema mappings by establishing connections between attributes in two schemas[3], which lacks

* This research is supported by the NSF of China under contract No.61272110 and No.61202036, the State Key Lab. of Software Engineering (SKLSE) under contract No.SKLSE20100829, Open Fund. of Shanghai Key Lab. of Intelligent Info. Processing under contract No.IIPL-2011-002, and the 111 Project under contract No.B07037.

expressiveness in presenting common data restructuring in integrating heterogeneous XML sources. Most studies on XML schema mapping languages usually adopt certain structural patterns, i.e., variables structurally composed with tags, to indicate that the data elements matching the patterns be in the structural relationships accordingly. The structural patterns can be roughly classified as path ones [4][5][6][7] and tree-like ones[8][9][10].

In these mapping languages, the variables in structural patterns are implicitly combined conjunctively and thus treated as a variable tuple \bar{x} . Thus the above formula is employed in XML schema mapping with little modification. However, this tuple-oriented schema mapping is insufficient to specify complex XML mapping requests involving hierarchical and heterogeneous data elements. The common solution is to use many mapping rules to specify various kinds of correspondences between variable tuples, which degrades the readability and the expressiveness of schema mapping specification.

In this paper, we propose a new approach to presenting XML schema mappings which deploys tree-like patterns composed with logical operators to present the requests on mapping hierarchical and heterogeneous data. In comparison with existing studies, our work contributes to presenting XML schema mapping in two sides: on one hand, we propose a novel pattern-based view named Conjunctive-Disjunctive Views (CDV) which uses conjunctive and disjunctive operators to compose patterns and the data elements accordingly. To our knowledge, it is the first attempt to introduce disjunctive operators in XML schema mappings[9]; b) on the other hand, we adopt a simple but expressive deductive mechanism to specify and resolve data transformation following certain restructuring rules. These restructuring rules are coherent with the conjunctive-disjunctive relationships among data elements indicated by CDVs, and they lay the foundation of building data hierarchy and handling heterogeneity.

The remainder of the paper is organized as follows. In Section 2, the design issues and the features of CDV are introduced. In Section 3, certain schema mappings with CDV are demonstrated with typical examples. Section 4 concludes the paper.

2 Conjunctive-Disjunctive View

The Conjunctive-Disjunctive View (CDV) is a pattern-based view for specifying logical and semantic relationships among data elements in an XML document fragment. The core of a CDV is a tree-like pattern named Conjunctive-Disjunctive Pattern (CDP) whose grammar is listed below.

As shown in Fig.1, a CDP (denoted by p) can be an atomic pattern (denoted by a), an element pattern (denoted by e) and a composite pattern. For an atomic pattern, the notation ϵ denotes a *null* value indicating the absence of an element, and s denotes a document fragment, e.g., a value or an element, indicating the existence of such a fragment under current context. An element pattern is composed of an optional location prefix, a label and an optional variable. An element pattern without location prefix denotes an element locating under the

$$\begin{aligned}
 a &:= \epsilon \mid s \\
 e &:= l \mid //l \mid l(x) \mid //l(x) \\
 p &:= a \mid e \mid e[p] \mid p^*p \mid p|p \mid p+p \mid \{p\}
 \end{aligned}$$

Fig. 1. The syntax of Conjunctive-Disjunctive Pattern

current context, and one with the prefix “//” denotes a descendant element. The label l denotes an element tag or an attribute name, and the variable x is used to bind the matching values. For example, matching the pattern $\text{title}(\$t)$ with an element $<\text{title}>v</\text{title}>$ would result in a variable binding “ $\$t \rightarrow v$ ”.

A composite CDP can be a tree pattern $e[p]$, a conjunctive pattern $p_1 * p_2$, an option pattern $p_1 | p_2$, a disjoint pattern $p_1 + p_2$ and a group pattern $\{p\}$. The tree pattern $e[p]$ is to match a document element where e matches the root tag and p matches the content of the element. The conjunctive pattern $p_1 * p_2$ indicates that the data respectively matching p_1 and p_2 be associated conjunctively. The option pattern $p_1 | p_2$ indicates that a matching element match either the pattern p_1 or the pattern p_2 . The disjoint pattern $p_1 + p_2$ indicates that the data respectively matching the pattern p_1 and p_2 should be disjoint from each other and be gathered disjunctively. The group pattern $\{p\}$ is actually a recursive option of disjoint patterns that $\{p\} = p \mid p+p \mid p+\dots+p \mid \dots$, and it indicates that the data matching the pattern p be disjoint from each other and are gathered as a set of independent elements.

Matching a composite CDP with a document fragment results in a compound variable binding composed of the element variable bindings in the similar structure of the CDP. CDV adopts an expression named **matching term** to denote the structure of these bindings, which exactly corresponds to the CDP structure. A matching term of a CDP is composed of the variables in a CDP with the correspondent conjunctive and disjunctive operators. For example, the CDP $\text{course}(\$c)[\{\text{//course}(\$cr)\} * \text{title}(\$t)]$ has the matching term $\$c * \{\$cr\} * \$t$, and would generate the binding like $\$c \rightarrow c_1 * (\$cr \rightarrow c_2 + \$cr \rightarrow c_3) * \$t \rightarrow t_1$.

A CDV is an extension of a CDP by embedding the constraints into the pattern to restrict the matching values of the pattern. A constraint is a predicate function which uses the matching term of the associated CDP as the actual argument. For example, the CDV ($\text{teacher}(\$x)$, $(\$x/\text{lastname}) = \text{"Li"}$) indicates that there be a teacher element bound to the variable $\$x$ which has the last name “ Li ”; $(\text{class}(&c1)[\text{teacher}(\$x1)] * \text{class}(&c2)[\text{teacher}(\$x2)], (\$x1 = \$x2) \text{ and } (&c1 != &c2))$ indicates that there be two different class elements, denoted as $\&c1 != \&c2$ where $\&x$ is a location variable, which are conjunctively combined and have common teacher ; $(\{(\text{teacher}(\$x), (\$x/\text{lastname}) = \text{"Li"})\}, \text{count}(\{\$x\}) > 2)$ indicates that there be some teacher elements which have the last name “ Li ” and are combined as a set having more than 2 members. Additionally, since the bindings of the sub-patterns of a disjunctive pattern are independent to each other, the CDVs based on the disjunctive views like $v_1|v_2$ or v_1+v_2 need not constraint.

XML schema mappings can naturally be presented as mappings between CDVs where a fundamental issue is to coherently transform the source data elements into the expected structure compatible with the target document. We thus deploy a deductive mechanism of the matching terms, using a set of rewriting rules to specify how one matching term can be restructured to another.

$$\begin{aligned}
 p^*/+/\mid p' \rightarrow p^*/+/\mid p & (\text{conj/disj}/\text{option-commutation}) \\
 p \rightarrow p^*p & (\text{conj-duplication}) \quad p^*p' \rightarrow p & (\text{conj-reduction}) \\
 p \leftrightarrow p^*\epsilon & (\text{conj-extension}) \quad p \rightarrow p \mid p' & (\text{option-extension}) \\
 p^*(p_1 +/\mid p_2) \rightarrow p^*p_1 +/\mid p^*p_2 & (\text{disj}/\text{option-distribution}) \\
 \{p^*p'\} \rightarrow \{f(p)\%*\{p^*p'\}\} & (\text{group-folding})^\dagger \\
 \dagger & \text{Grouping the set into a collection by the distinct values of } f(p).
 \end{aligned}$$

Fig. 2. Restructuring rules for matching terms

The rules in Fig.2 are designed on the purposes of: a) coherently maintaining the conjunctive-disjunctive relationships among the data elements during data transformation; and b) flexibly reorganizing the data structure to be compliant to the target document. This idea comes from our previous study on XML query and for the further theoretical properties please refer to [11] .

3 Presenting XML Schema Mappings with CDV

Based on CDVs, a XML schema mapping can be presented following a formula $\varphi_S(mt_s(\bar{x})) \rightarrow \exists \bar{y}. \psi_T(mt_t(\bar{x}, \bar{y}))$ where $\varphi_S(mt_s(\bar{x}))$ denotes the source CDVs over the matching term $mt_s(\bar{x})$ and $\psi_T(mt_t(\bar{x}, \bar{y}))$ denotes a target CDV over the matching term $mt_t(\bar{x}, \bar{y})$. Here the variables \bar{y} are fresh ones denoting the values in the target CDV, and the variables \bar{x} are bound ones originated from the source views. Generally, a CDV-based mapping is specified as the expression “**from** source-view **to** target-view”. A source view is specified as a source document or a schema with a CDV, indicating how the data from local sources be mapped to the target view. The structural and semantic requirements of the target document are defined by the target views.

Currently CDV only works on the documents with DTD schemas and thus only allows some kinds of simple constraints indicating common features of DTD schemas. The available predicates in constraints include filtering elements with constants and one-argument functions, e.g., $\$x/last-name = "Li"$, judging value equivalence, e.g., $\$a = \b , comparing locations, e.g., $\&a > \&b$ ($\&a$ locates behind $\&b$) or $\&a << \&b$ ($\&a$ precedes $\&b$), comparing sets, $\{\$a\} \leq \{\$b\}$ and judging element in set, e.g., $\$a$ in $\{\$b\}$. Additionally, certain pragmatic paradigms for presenting common mappings are provided to facilitate processing where some special constraints are allowed.

Fig.3 shows a simple case of XML integration. The course and the class documents respectively following the *s1.dtd* and *s2.dtd* schemas are to be transformed and integrated into a target document following *t.dtd*. There is a “pre-required”

<p>s1.dtd: $r \rightarrow \text{course}^*$</p> <p style="margin-left: 20px;">$\text{course} \rightarrow \text{cno}, \text{title}, \text{prereq}$</p> <p style="margin-left: 20px;">$\text{prereq} \rightarrow \text{course}^*$</p> <p>s2.dtd: $r \rightarrow \text{class}$</p> <p style="margin-left: 20px;">$\text{class} \rightarrow \text{courseno} \text{classno},$</p> <p style="margin-left: 20px;">$\text{title}?, \text{year}, \text{teacher}$</p>	<p>t.dtd: $r \rightarrow \text{course}^*$</p> <p style="margin-left: 20px;">$\text{course} \rightarrow \text{cno}, \text{title}+,$</p> <p style="margin-left: 20px;">$\text{class}^*, \text{reqby}$</p> <p style="margin-left: 20px;">$\text{reqby} \rightarrow \text{cno}^*$</p> <p style="margin-left: 20px;">$\text{class} \rightarrow \text{year}, \text{teacher}$</p>
--	---

Fig. 3. A case of XML schema mapping

(*prereq*) relationship between courses indicating that a course be studied after certain courses. During the integration, the recursively-defined course elements in *s1.dtd* should be flattened as a list of course elements in *t.dtd*, and the *pre-required* relationship should be interpreted to the “*required-by*” (*reqby*) relationship. In this list, the sub-element “*cno*” is the key for each course element, and a course should locate before its “*reqby*” courses. The class information in *s2.dtd* should be grouped by “*courseno*” or “*classno*” and then combined into *t.dtd* accordingly. Here the content of “*classno*” should be changed to a “*courseno*” by a function *ch()*. These schema mapping requests can be presented with the mapping CDVs in the following examples.

Example 1. Extract the course information from the original course elements into a temporary document containing the fragments compliant to the target document.

```
from (s1.dtd){//course[cno($n)*
    prereq[{course[cno($n1)*title($t)]}|null]}
to (tmp1){course[cno($n1)*({title($t1)}*reqby[{cno($n%)}],{$t%}<={$t1})]}
```

Example 1 transforms a recursive data hierarchy into a flat set. In the source CDV all the pairs of courses constituting the pre-required relation are extracted as $\{\$n*\$n1\}$, and then set is folded to $\{\$n1\%*\$n\%$ indicating the required-by relation in the target schema.

Example 2. Group the courses in the class elements by its “*courseno*” or “*classno*” to generate a temporary document containing the fragments compliant to the target document.

```
from (s2.dtd) {class[(courseno($n1)|classno($n2))*({title($t)}|null)
    *year($y)*teacher($x)]
to (tmp2)({(course[cno((\$n1|ch(\$n2))%)*({title($t1)}|null)*
    [class[year($y)*teacher($x)]]]}, {$t%}<={$t1}))}
```

In this example, the source term $\{(\$n1|\$n2)*(\$t|null)*\$y*\$x\}$ is restructured to $\{(\$n1|\$n2)\%*\{(\$n1|\$n2)*(\$t|null)*\$y*\$x\}\}$, and further to $\{(\$n1|\$n2)\%*(\{$t\}|null)*\{\$y*\$x\}\}$ in the target CDV. The target CDV directly uses $(\$n1|\$n2)\%$ and $\{\$y*\$x\}$ in CDP, and introduces a fresh variable $\$t1$ with the constraint $\{$t\}<=\{$t1\}$ to indicate that all the bound values of $\$t$ should be used to instantiate $\$t1$. Here the set inclusion rather than set equivalence is used because the mapping just constructs partial information of the target document.

Example 3. Join the course elements from two temporary documents into the target document.

```
from ((tmp1){course[cno($n2)*{title($t2)}*reqby($r)]], $n1=$n2)*
      (tmp2){course[cno($n1)*{title($t1)}*({class($c)}|null)]})
to (tgt@t.dtd){course[cno($n1)*{title($t)}*({class($c)}|null)*
                 reqby($r)], {$t}=union({$t1}, {$t2})}
```

4 Conclusion

XML schema mapping is an important and interesting topic in XML integration and exchange. Existing studies often follow the conventional approach to presenting the mapping as the dependence between variable tuples. In this paper, We introduced a pattern-based approach named CDV to present XML schema mapping as the correspondence between different but compatible structures of variables composed of logical operators. CDV adopts disjunctive as well as conjunctive operators to compose sub-patterns and the associating semantic constraints, and is convenient to present complex mapping requests on hierarchical and heterogeneous data. Due to the tight space constraints, only the basic ideas of designing CDV are illustrated in this paper. For more details of the theoretical issues of CDV and the examples, please refer to the full report[12].

References

1. Lenzerini, M.: Data integration: a theoretical perspective. In: ACM PODS 2002, pp. 233–246 (2002)
2. Kolaitis, P.: Schema mappings, data exchange, and metadata management. In: ACM PODS 2005, pp. 61–75 (2005)
3. Barbosa, D., Freire, J., Mendelzon, A.: Designing information-preserving mapping schemes for XML. In: VLDB 2005, pp. 109–120 (2005)
4. Aguilera, V., Cluet, S., Milo, T., Veltri, P., et al.: Views in a large-scale XML repository. VLDB J. 11(3), 238–255 (2002)
5. Fan, W., Garofalakis, M.N., Xiong, M., Jia, X.: Composable XML integration grammars. In: CIKM 2004, pp. 2–11 (2004)
6. Poggi, A., Abiteboul, S.: XML Data Integration with Identification. In: Bierman, G., Koch, C. (eds.) DBPL 2005. LNCS, vol. 3774, pp. 106–121. Springer, Heidelberg (2005)
7. Fan, W., Bohannon, P.: Information preserving XML schema embedding. ACM TODS 33(1) (2008)
8. Arenas, M., Libkin, L.: XML data exchange: consistency and query answering. Journal of ACM 55(2) (2008)
9. Amano, S., Libkin, L., Murlak, F.: XML Schema Mappings. In: PODS 2009, pp. 33–42 (2009)
10. Bonifati, A., Chang, E., Ho, T., Lakshmanan, L., et al.: Schema mapping and query translation in heterogeneous P2P XML databases. VLDB J. 19, 231–256 (2010)
11. Li, X., Liu, M.: XTQ: A Declarative Functional XML Query Language, Tech. Report (2012), <http://www.sklse.org/xtq>
12. Li, X., Liu, M.: CDV: A Conjunctive-Disjunctive View for XML Schema Mapping. Tech. Report (2012), <http://www.sklse.org/xtq>

An ETL Framework for Online Analytical Processing of Linked Open Data

Hiroyuki Inoue¹, Toshiyuki Amagasa², and Hiroyuki Kitagawa²

¹ Graduate School of Systems and Information Engineering, University of Tsukuba

² Faculty of Engineering, Information and Systems, University of Tsukuba

1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

inohiro@kde.cs.tsukuba.ac.jp, {amagasa,kitagawa}@cs.tsukuba.ac.jp

Abstract. Growing amount of data are being published online in machine-readable formats, and LOD (Linked Open Data) has emerged as a way to share such data across Web resources. Since LOD data often contain numerical data, such as statistics, there is a growing demand to make OLAP (Online Analytical Processing) analysis over such data. To make it possible to apply off-the-shelf OLAP systems for analyzing LOD data, we propose a framework to streamline the Extract, Transform, and Load (ETL) process from LOD to multidimensional data models for OLAP. Unlike other related approaches, our framework does not require RDF vocabularies dedicated for specifying multidimensional model for OLAP. Instead, given an LOD dataset, we exploit the relationships among entities and external information in the referenced LOD to generate an OLAP data model. In a case study, we demonstrate that our framework can extract OLAP data models from different kinds of real LOD datasets.

Keywords: Linked Open Data, OLAP, ETL, Multidimensional Model.

1 Introduction

Linked Open Data (LOD) [2] is an increasingly popular method that allows organizations to publish their data in machine-readable format, in particular using the Resource Description Framework (RDF) [3]. According to the State of the LOD Cloud¹, over 31.6 billion triples in 295 datasets were published as of September 2011, covering life science fields, remote sensors, and governments, among others.

Most LOD datasets comprise not only text data, but also numerical data. For example, [9] is to publish statistical data about Eurostat² and Linked Sensor Data [13] is to publish sensor readings such as temperature, humidity, etc. For this reason, to make the best use of this type of data, it is vital to be able to retrieve necessary information from massive LOD datasets and apply potentially complex analytical processing to them [6, 10].

Online analytical processing (OLAP) [4, 8] has been used in a wide spectrum of applications as a powerful means of performing online analysis and reporting over databases containing numerical attributes. For this reason, OLAP is one of the easiest and natural ways to analytically process LOD datasets. Several works have applied

¹ State of the LOD Cloud.

<http://wifo5-03.informatik.uni-mannheim.de/lodcloud/state/>

² Eurostat. <http://ec.europa.eu/eurostat/>

OLAP to LOD [6, 7, 10]. One approach involves transforming LOD datasets to relational tables, and performing OLAP analysis using existing OLAP systems [10]. Alternatively, one may directly perform OLAP-like analysis over LOD datasets [6]. Since there are many off-the-shelf systems that support OLAP analysis, the former approach has certain advantages.

When applying Extract, Transform, and Load (ETL) [8] processes to LOD datasets for subsequent OLAP analysis, there are three technical challenges: 1) designing the target star (or snowflake) schema consisting of fact and dimension tables, 2) extracting concept hierarchies that are associated with the dimension tables, and 3) determining how best to deal with external LOD datasets that can be accessed through URL references.

In this paper we propose a general ETL framework for LOD datasets. The basic features are: 1) it can deal with any LOD datasets regardless of the usage of dedicated OLAP cube vocabularies, while other methods require such vocabularies like RDF Data Cube Vocabulary (QB) [5]; and 2) it can exploit inherent semantic hierarchies in the dataset, as well as those extracted from external LOD datasets such as GeoNames [14] and DBpedia [1]. In the case study using real LOD datasets, we show that the proposed framework can successfully extract multidimensional data model for OLAP dataset.

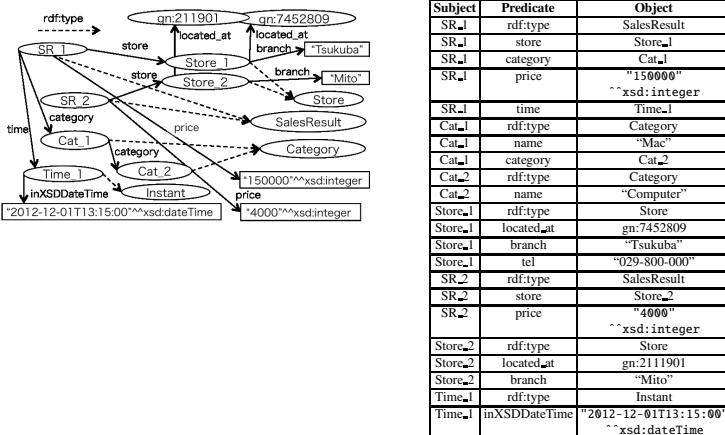
2 Proposed Framework

2.1 Storing RDF Data

Given an LOD data, they are first stored in intermediate storage. The storage schema is designed in such a way that it is independent of the schemas used in the OLAP system. To this end, we basically use the Property Table (PT) approach [15, 16], where for each distinct type (`rdf:type`), a table is created with a number of attributes corresponding to RDF properties. Specifically, we take the following steps:

Type-partitioned Triple Store (TPTS). After obtaining the RDF data represented as a collection of triples, we partition them according to the type (`rdf:type`) of the subject, resulting in a TPTS. More precisely, to maintain the distinction between resource and literal, we annotate the data type information in the object column in the TPTS, such as `xsd:integer`, `xsd:dateTime`, etc. Figure 1 shows an example of RDF data. By using TPTS, it can be represented using several tables as shown in Table 1.

Property Table Generation. Having extracted a TPTS, we convert it to a set of Property Tables (PTs). Basically, we create a PT for each partition in the TPTS, because the subjects in the same partition are of the same type. The primary key of the table is the subject URI. For other attributes, we first enumerate the set of properties appearing in the partition by scanning the table, and use this information to create the set of attributes in the PT being generated. We carry out the same process for each partition, yielding the PTs in Table 2.

**Fig. 1.** RDF data and triple representation

Cardinality Estimation of Foreign Keys. In order to generate a star schema it is important to maintain the cardinality information for a foreign key, but this information is not explicitly provided. For this reason, we estimate the cardinality from the instance (in other words, from each record). For each foreign key, we check the occurrences in the source and referenced tables, and estimate the cardinality (1:1, 1:N, N:1, or M:N). This information is also maintained in the framework (Table 2(d)).

Table 1. Type-partitioned triple store for SalesResult, Store, and Instant

(a) rdf:type: SalesResult				(b) rdf:type: Store				(c) rdf:type: Category			
Subject	Predicate	Object	Data Type	Subject	Predicate	Object	Data Type	Subject	Predicate	Object	Data Type
SR ₁	store	Store ₁	Resource	Store ₁	located_at	gn:7452809	Geonames	Cat ₁	name	"Mac"	String
SR ₁	category	Cat ₁	Resource	Store ₁	branch	"Tsukuba"	String	Cat ₁	category	Cat ₂	Resource
SR ₁	price	150000	Integer	Store ₁	tel	"029-800-000"	String	Cat ₂	name	"Computer"	String
SR ₁	time	Time ₁	Resource	Store ₂	located_at	gn:2111901	Geonames	Store ₂	branch	"Mito"	String
SR ₂	store	Store ₂	Resource								
SR ₂	price	4000	Integer								

2.2 Dimension Induction

After storing the RDF data in PTs, the system shows the PTs and the respective attributes to the user. The user then selects one numerical attribute in a table, which is used as the measure in the fact table in the subsequent process.

From the specified fact table, the system identifies the possible dimensions (except for the measure) that can be found in the attributes of the fact table or those in tables referenced by foreign keys. In the following, three typical cases are discussed.

Using Literals. In this case, an attribute in the fact table, or one in another table that can be reached by following a 1:1 or an N:1 reference, can be directly associated to the concept hierarchy. Timestamp and location are the most popular examples, i.e.,

Table 2. Property table representation

(a) rdf:type: SalesResult	(b) rdf:type: Store																																		
<table border="1"> <thead> <tr> <th>Subject</th><th>store</th><th>category</th><th>price</th><th>time</th></tr> </thead> <tbody> <tr> <td>SR₁</td><td>Store₁</td><td>Cat₁</td><td>15000</td><td>Time₁</td></tr> <tr> <td>SR₂</td><td>Store₂</td><td>null</td><td>4000</td><td>null</td></tr> </tbody> </table>	Subject	store	category	price	time	SR ₁	Store ₁	Cat ₁	15000	Time ₁	SR ₂	Store ₂	null	4000	null	<table border="1"> <thead> <tr> <th>Subject</th><th>located_at</th><th>branch</th><th>tel</th></tr> </thead> <tbody> <tr> <td>Store₁</td><td>gn:7452809</td><td>"Tsukuba"</td><td>"029-800-000"</td></tr> <tr> <td>Store₂</td><td>gn:2111901</td><td>"Mito"</td><td>null</td></tr> </tbody> </table>	Subject	located_at	branch	tel	Store ₁	gn:7452809	"Tsukuba"	"029-800-000"	Store ₂	gn:2111901	"Mito"	null							
Subject	store	category	price	time																															
SR ₁	Store ₁	Cat ₁	15000	Time ₁																															
SR ₂	Store ₂	null	4000	null																															
Subject	located_at	branch	tel																																
Store ₁	gn:7452809	"Tsukuba"	"029-800-000"																																
Store ₂	gn:2111901	"Mito"	null																																
(c) rdf:type: Category	(d) Relationships among PTs.																																		
<table border="1"> <thead> <tr> <th>Subject</th><th>name</th><th>category</th></tr> </thead> <tbody> <tr> <td>Cat₁</td><td>"Mac"</td><td>Cat₂</td></tr> <tr> <td>Cat₂</td><td>"Computer"</td><td>null</td></tr> </tbody> </table>	Subject	name	category	Cat ₁	"Mac"	Cat ₂	Cat ₂	"Computer"	null	<table border="1"> <thead> <tr> <th>table</th><th>column</th><th>f.table</th><th>f.column</th><th>is_onezone</th></tr> </thead> <tbody> <tr> <td>SalesResult</td><td>store</td><td>Store</td><td>Subject</td><td>false</td></tr> <tr> <td>SalesResult</td><td>time</td><td>Instant</td><td>Subject</td><td>false</td></tr> <tr> <td>Store</td><td>located_at</td><td>GeoNames</td><td>Subject</td><td>false</td></tr> <tr> <td>Category</td><td>category</td><td>Category</td><td>Subject</td><td>false</td></tr> </tbody> </table>	table	column	f.table	f.column	is_onezone	SalesResult	store	Store	Subject	false	SalesResult	time	Instant	Subject	false	Store	located_at	GeoNames	Subject	false	Category	category	Category	Subject	false
Subject	name	category																																	
Cat ₁	"Mac"	Cat ₂																																	
Cat ₂	"Computer"	null																																	
table	column	f.table	f.column	is_onezone																															
SalesResult	store	Store	Subject	false																															
SalesResult	time	Instant	Subject	false																															
Store	located_at	GeoNames	Subject	false																															
Category	category	Category	Subject	false																															

the former can be associated with a hierarchy containing day, week, month, and year, while the latter can be associated with one containing city, state, and country.

Exploiting Inherent Hierarchy. In many cases, the RDF data being processed represent hierarchical structures. In such cases, the hierarchical structure is represented as a triple, such as `rdfs:subClassOf`, `rdfs:subPropertyOf`, etc. In our framework, such a structure is represented as a set of self references in a PT. By following them, we can generate a dimension table corresponding to the concept hierarchy.

Exploiting External Information. A piece of LOD data is likely to contain references to other LOD datasets. Some of the existing LOD datasets are quite popular and tend to collect many references. Examples include GeoNames [14] for location information and DBpedia [1] for encyclopedic information. In addition, there are various well-known ontologies, such as “Time Ontology”³, “Timeline Ontology”⁴, and “WG84 Geo Position Ontology”⁵. Such LOD datasets or ontologies are popular and are frequently referred to. For these, it is useful to extract the concept hierarchies beforehand and maintain them as the known dimension tables.

2.3 Schema Generation

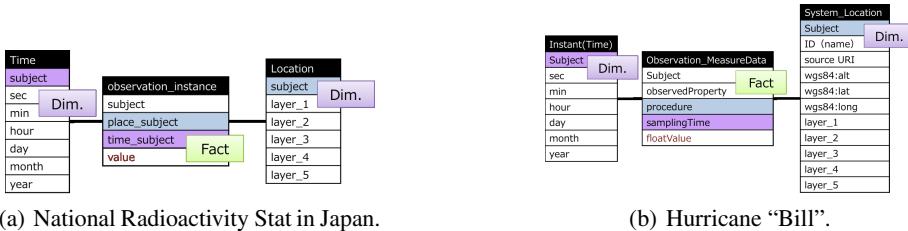
Now we are ready to generate the relational schema for the OLAP system. Basically, the system shows the list of possible dimension tables to the user, and he choose some of them to be used in the OLAP analysis. Having specified the list of dimension tables, we generate the concrete schema definition for the fact table and the selected dimension tables. More precisely, we generate projected tables by omitting unused attributes, and merge such tables that are associated by 1:1 relationships for the sake of schema simplicity.

Regarding data loading, there are several possibilities. If the database storing the PTs is shared with the OLAP system, it is possible to create the fact and dimension tables as the view of PTs. If the database is not shared with the OLAP system, the data records must be dumped and subsequently loaded into the OLAP system.

³ The Time Ontology in OWL. <http://www.w3.org/TR/owl-time/>

⁴ The Timeline Ontology. <http://motools.sourceforge.net/timeline/>

⁵ WG84 Geo Position Ontology. <http://www.w3.org/2003/01/geo/>

**Fig. 2.** The generated schemas

3 Case Study

To show the applicability of the proposed scheme, we have applied the proposed framework to two real LOD datasets.

Datasets. The first dataset is the “National Radioactivity Stat as Linked Data”⁶. This dataset was created experimentally by MEXT (Ministry of Education, Culture, Sports, Science and Technology) of Japan by converting a non-LOD dataset⁷ to LOD. It provides the radioactivity monitoring results from March 16, 2011 to March 15, 2012 at 47 prefectures in Japan.

Another dataset is Linked Sensor Data [13], which is a dataset containing metadata from 9,757 weather observatories in the U.S. These metadata cover ID, coordinates described using WG84 Geo Position Ontology, location information consisting of links to GeoNames, and so on. Additionally, Linked Observation Data contains a dataset of hurricane-related observations linking to the Linked Sensor Data. Each observation in the Linked Observation Data includes a numerical measurement, the type of observation (such as rainfall, wind speed, humidity, and so on), observation time, and observation site described using the Linked Sensor Data.

Results. Figures 2(a) and 2(b) show the resulting schemas. By specifying the fact in each schema, we can successfully extract dimensions, and generated the schema for OLAP system.

4 Related Work

Kampgen et al. [10] proposed an ETL process for Linked Data that used RDF Data Cube Vocabulary (QB) [5]. Other approaches apply analytical processing to RDF resources using RDF/OWL [11] ontologies. Niemi et al. [12] defined the OLAP Core ontology and proposed a method combining RDF, which is made up of distributed and heterogeneous sources, with OLAP. Using a different approach, Etcheverry et al. [6] defined OLAP operations for an RDF data model, and proposed an OpenCube Vocabulary

⁶ National Radioactivity Stat as Linked Data. <http://kanzaki.com/works/2011/stat/ra/>

⁷ Monitoring information of environmental radioactivity level.

<http://radioactivity.nsr.go.jp/ja/>

in order to apply OLAP-like operations such as roll-up and slice with the aggregation function of SPARQL.

5 Conclusion

In this paper, we proposed an ETL framework for the analytical processing of LOD. To do this we mapped resources and the relationships among them to relational tables and their reference relations. Once this type of mapping is done, we are able to extract concept hierarchies from the dataset, exploiting either literals, relations, or external information. We also demonstrated the prototype of our framework and presented two case studies using real LOD datasets to prove the framework's utility. In the future, we plan to extend the proposed framework to deal with more general LOD data, and apply it to various datasets.

Acknowledgements. This work was supported by JSPS Grant-in-Aid for Young Scientists (B) (#23700102).

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A Nucleus for a Web of Open Data. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
2. Berners-Lee, T.: Linked Data - Design Issues, <http://www.w3.org/DesignIssues/LinkedData.html>
3. Carroll, J.J., Klyne, G.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C recommendation, W3C (February 2004), <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>
4. Codd, E., Codd, S., Salley, C., Codd & Date, Inc.: Providing OLAP (On-line Analytical Processing) to User-analysts: An IT Mandate. Codd & Associates (1993)
5. Cyganiak, R., Reynolds, D.: The RDF Data Cube Vocabulary. W3C working draft, W3C (April 2012), <http://www.w3.org/TR/vocab-data-cube/>
6. Etcheverry, L., Vaisman, A.A.: Enhancing OLAP Analysis with Web Cubes. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 469–483. Springer, Heidelberg (2012)
7. Etcheverry, L., Vaisman, A.A.: QB4OLAP: A Vocabulary for OLAP Cubes on the Semantic Web. In: COLD. CEUR Workshop Proceedings, vol. 905. CEUR-WS.org (2012)
8. Han, J., Kamber, M.: Data Warehouse and OLAP Technology: An Overview. In: Data Mining: Concepts and Techniques, 2nd edn., pp. 105–156. Morgan Kaufmann (2006)
9. Iqbal, A., Capadisli, S., Cyganiak, R., Hausenblas, M.: Eurostat - Linked Data, <http://eurostat.linked-statistics.org/>
10. Kämpgen, B., Harth, A.: Transforming statistical linked data for use in OLAP systems. In: I-SEMANTICS. ACM International Conference Proceeding Series, pp. 33–40. ACM (2011)
11. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. W3C recommendation, W3C (February 2004), <http://www.w3.org/TR/2004/REC-owl-features-20040210/>

12. Niemi, T., Toivonen, S., Niinimäki, M., Nummenmaa, J.: Ontologies with Semantic Web/Grid in Data Integration for OLAP. *Int. J. Semantic Web Inf. Syst.* 3(4), 25–49 (2007)
13. Patni, H., Henson, C.A., Sheth, A.P.: Linked Sensor Data. In: CTS, pp. 362–370 (2010)
14. Vatant, B., Wick, M.: GeoNames Ontology, <http://www.geonames.org/ontology/>
15. Wilkinson, K., Sayers, C., Kuno, H.A., Reynolds, D.: Efficient RDF Storage and Retrieval in Jena2. In: SWDB, pp. 131–150 (2003)
16. Wilkinson, K.: Jena Property Table Implementation. In: SSWS (2006)

A Framework for Analyzing Monetary Cost of Database Systems in the Cloud

Changbing Chen and Bingsheng He

School of Computer Engineering
Nanyang Technological University, Singapore

Abstract. In this paper, we propose to develop a framework to analyze the monetary cost of running database systems in the public cloud. The framework offers guidelines and methodologies in analyzing and estimating monetary cost of database systems. It consists of multiple components including categorizing database performance tuning knobs, benchmarking the price/performance of computation resources offered by the cloud provider, and building a monetary cost model. As a case study of our proposed framework, we conduct an in-depth study on two popular open-source database systems with respect to two cloud providers. We find that evaluating a query spans a wide range of monetary costs (with a difference up to 91%), and the experimental results demonstrate the accuracy of our monetary cost estimation.

Keywords: Database Systems, Cloud Computing, Performance, Monetary Cost, Framework.

1 Introduction

In the era of cloud computing, computation can be traded as a utility sold and bought according to predefined price schemes. Under *pay-as-you-go* charging, the monetary cost of running a database system in the cloud becomes an explicitly measurable metric. The *pay-as-you-go* feature is a clear distinction from previous price-oriented studies, for example, PennySort [3] and TPC-H including costs on hardware, software and shipping. Like the performance (in terms of throughput or response time) that has long been the core metric for system design and optimizations, the monetary cost has become an important metric for system design and optimizations [13,8].

To understand and optimize the monetary cost of running database systems in the cloud, we are particularly interested in the following questions:

- Where does the money go? We hope to identify the major cost component(s) in the total monetary cost.
- What is the correlation between performance and monetary cost of running database systems in the cloud?
- Current price schemes are mostly based on resource consumptions, with differential prices on virtual machine types. How do price schemes affect the system optimizations for monetary cost in the future cloud environment?
- What are the opportunities to reduce the monetary cost given certain workloads?

It is a non-trivial task to answer the above questions. The monetary cost of running database systems in the cloud involves various complicated and often intertwined factors involving cloud providers (including price definitions as well as the virtual machine performance), and users (including how database systems are tuned and optimized). All these factors are able to significantly affect the monetary cost. Specifically, providers offer virtual machines with different capabilities at different prices. It is not clear about the monetary cost of running the same system on these offerings. Moreover, there is not yet a standard on the price scheme among different cloud providers. Clearly, different price schemes directly result in different monetary costs, leading to the differences in monetary cost. Even within a single price scheme, different virtual machines are charged at different prices; resource components (such as I/O, CPU and memory) can have different prices at different time periods (such as option pricing in Amazon). There have been only a few preliminary studies in understanding and optimizing the monetary cost of running database systems in the cloud [13,9]. Also, few studies systematically explore the interplay among database systems, prices and cloud offerings.

In this paper, we develop a framework to guide the study on the interplay among performance tunings, price structures and cloud offerings. In the framework, we develop methodologies in the following components of analyzing the monetary cost: 1) defining the space of prices under study, 2) assembling (micro) benchmarks for assessing the price and performance of resource components of virtual machines, 3) categorizing the tuning knobs within database systems with respect to the relationship between performance and monetary cost, and 4) developing a monetary cost model to estimate the monetary cost of query processing. Those components are designed as guidelines for analyzing the monetary cost of database systems in the cloud.

We evaluate our framework by running two popular open-source DBMSs (PostgreSQL and MySQL) on two different cloud providers, Amazon and Rackspace. Through the extensive study, we find that virtual machine selections and price-aware optimizations are key factors for reducing the monetary cost. Additionally, the experiments show that our monetary cost model accurately predicts the monetary cost with different virtual machine offerings and prices. That demonstrates the effectiveness of the proposed framework.

Organization. The remainder of this paper is organized as follows. We review the related work in Section 2. Section 3 presents the framework design, followed by the experimental results in Section 4. Finally, we conclude this paper in Section 5.

2 Related Work

There have been a few studies on evaluating and optimizing user expenses in the cloud. Economics, particularly prices, have significant impact on cloud system designs [13,5]. Palankar et al. [9] studied the cost, availability and performance on Amazon S3 services, in the context of data intensive scientific applications. Kossmann et al. [8] studied the performance and cost of transactional workloads for different cloud providers. Tak et al. [11] studied different deployment choices for transactional workloads. Killapi et al. [7] proposed optimizations for virtual machine selections to minimize the monetary cost of data flow programs. Assuncao et al. [2] evaluated the cost performance

of different scheduling strategies to combine a private (self-owned) cloud with public clouds. Compared with existing work, this study focuses on the interplay among database systems, prices and cloud offerings, and develops a monetary cost model for monetary cost optimizations. A number of studies [13,10,4] have been conducted to understand the performance variance on Amazon. We observed similar variance in the monetary cost, and our study covers the offerings from different cloud providers, and more virtual machine types on Amazon.

3 Framework Design

In order to analyze the monetary cost of database systems in the cloud, we develop a framework to guide the analysis of different aspects related to monetary cost. Particularly, the framework consists of the following components: 1) defining the space of prices, 2) assembling a series of micro benchmarks for assessing the price and performance of resource components of virtual machines, 3) categorizing the tuning knobs within database systems with respect to the relationship between performance and monetary cost, and 4) developing a monetary cost model by extending the performance model within database system. In the remainder of this section, we present the detailed design of each component.

3.1 Space of Prices

We observe *spatial* and *temporal* features of price definitions in the cloud. The spatial feature means different price definitions for different virtual machine instances and different cloud providers. The temporal feature means the temporally varying prices for the same virtual machine type, such as option pricing on Amazon. One example of spatial features is differential pricing for virtual machines in different regions of the world. Even for the same virtual machine, the reservation price is lower than that of on-demand virtual machines. Option pricing in Amazon is a temporal pricing feature. When the option price determined by supply/demand is lower than the user bid price, the virtual machine is granted to the user at the option price.

For the fair comparison on different virtual machines, we consider the offerings with the same operating system (Linux in our study). Moreover, the absolute prices are not a direct motivation for investigating the monetary cost of system optimizations. The interplay between the database system optimization and the price is on the *relative* prices among different resource consumptions, the *relative* prices for the same resource within the same provider or from different cloud providers and the *relative* prices on different temporal periods. We intentionally vary the price structures for sensitivity evaluations on the monetary cost.

3.2 Micro Benchmarks

We assemble the following micro benchmarks to measure the monetary cost of CPU and memory systems. Particularly, we use the Unix Benchmark Utility (Ubench) [12] to measure CPU capability and main memory performance. For I/O, we use the IOzone benchmark [6] to measure the sequential and random read performance on the persistent storage.

3.3 Categorizing Tuning Knobs

Many performance tuning knobs could have been added to DBMSs for performance evaluation. They cover different aspects of data management tasks, from query planning and execution to system wide configurations. Our studies identify three categories of tuning knobs, according to resource consumption based price schemes: (Category A) the knobs directly reduce the consumption of a resource component without increasing the consumption of other resource components (e.g., the buffer size of the DBMS), and (Category B) the knobs have the tradeoff among multiple resource components: reducing the consumption of a resource component but increasing the consumption of the other resource components. Tuning knobs in Category B require careful investigations on existing tradeoffs, which exist in various query optimizations such as access methods (e.g., sequential scans vs. index scans), and compression techniques. Beyond the existing tuning knobs for a specific virtual machine, we have other tuning knobs in the cloud, i.e., selecting different cloud providers and different virtual machine types from a specific provider (Category C). In this study, we investigate some typical tuning knobs that are common for PostgreSQL and MySQL, as shown in Table 1.

Table 1. Example tunings in each category

Category	Tunings
A	<ul style="list-style-type: none"> • tuning the buffer size;
B	<ul style="list-style-type: none"> • database compressions; • auxiliary structure (e.g., index and materialized view);
C	<ul style="list-style-type: none"> • selections of different virtual machines from the same provider; • selections of suitable cloud providers;

3.4 Monetary Cost Model

Since existing cost models in DBMSs are limited to query execution time, we develop a monetary cost model to facilitate virtual machine selections and price-aware tuning knobs. A natural direction is to adapt the existing cost model in DBMS with the awareness of virtual machines and prices. We have implemented the monetary cost model in both PostgreSQL and MySQL. Since their adaptation and experimental results are similar, we present the adaptation in details with PostgreSQL as an example.

We develop the monetary cost model in two steps, introducing the awareness of virtual machine performance differences and prices into the existing timing cost model of the DBMS.

First, we introduce the awareness of the performance differences of virtual machine offerings, and obtain an enhanced timing cost model. As demonstrated in Figure 2, different virtual machine types have quite different CPU, memory and I/O performance. The existing timing cost model of the DBMS uses some constants as the unit cost for calculating the query execution time. Figure 1 shows a fraction of the source code for the cost estimation of a sequential scan on a relation in PostgreSQL. Two constant parameters, “spc_seq_page_cost” and “cpu_tuple_cost”, are used to represent the I/O

and CPU unit costs per tuple in a sequential scan. We use some simple experiments to calibrate those unit costs in the target virtual machine types. Next, the calibrated values are plugged into the timing cost model. Thus, the timing cost model is able to predict the timing cost on different virtual machine types.

```
//in costsizes.c in the PostgreSQL code base
00172 cost_sequential(Path *path, PlannerInfo *root,
00173 RelOptInfo *baserel)
00174 {
...
/* Introduce virtual machine
   awareness to unit costs
   */
00193 * disk costs
00194 */
00195 run_cost += spc_seq_page_cost * baserel->pages;
00196
00197 /* CPU costs */
00198 startup_cost += baserel->baserestrictcost.startup;
00199 cpu_per_tuple = cpu_tuple_cost + baserel->baserestrictcost.per_tuple;
00200 run_cost += cpu_per_tuple * baserel->tuples;
...
00203 path->total_cost = startup_cost + run_cost;
00204 }
```

Fig. 1. An illustration of building monetary cost estimation: estimations for a sequential scan on a relation in PostgreSQL

Second, we introduce the awareness of price structures into the enhanced timing cost model. The enhanced timing cost model is able to predict the query execution time (denoted as T) on a certain virtual machine type. Moreover, from the timing cost estimation, we can obtain the estimated number of I/O operations incurred during query execution (denoted as $\#IO$). Figure 1 illustrates that “`baserel->pages`” and “`total_cost`” are used as estimations in PostgreSQL for $\#IO$ and T , respectively. Therefore, we derive the monetary cost model in Eq. 1, by summing up the execution, storage and I/O cost components. The price structure ($p_{execution}$, $p_{storage}$ and p_{io}) is defined as the prices for virtual machine execution, storage and I/O operations, respectively. The database storage size, S , is obtained from the database catalog.

$$MC = p_{execution} \times T + p_{storage} \times S \times T + p_{io} \times \#IO \quad (1)$$

4 Case Studies

In this section, we present case studies of the proposed framework by running PostgreSQL and MySQL on Amazon and Rackspace.

4.1 Experimental Setup

SUT (System Under Test). The SUT (PostgreSQL or MySQL) runs on a virtual machine in a public cloud. The database data are stored in the persistent storage offered in the cloud, e.g., EBS/RAID 0 in Amazon and the persistent storage of *cloud server* in Rackspace. Therefore, database data is persistent, even after the virtual machine is turned off.

Table 2. Instances on Amazon used in our experiments (Price: \$ per hour, July 15, 2012, Linux, US-N. Virginia)

Name	Disk (GB)	CPU (CU)	IO	RAM (GB)	Price	Option price
A1	0	≤ 2	Low	0.6	0.02	0.008
A2	160	1×1	Moderate	1.7	0.085	0.035
A3	350	2×2.5	Moderate	1.7	0.17	0.06
A4	850	2×2	Moderate	7.5	0.34	0.15
A5	1,690	8×2	High	15	0.68	0.24

Table 3. Linux virtual machines on Rackspace used in our experiments (July 15, 2012)

Name	Disk (GB)	RAM (GB)	Price (\$ per hour)
R1	80	2	0.12
R2	160	4	0.24
R3	320	8	0.48
R4	620	15.5	0.96

TPC-H Setup. We chose TPC-H as the benchmark workloads, since these queries represent commonly used data warehousing workloads. Moreover, they vary in complexity, which exercises different components including CPU and I/O in the virtual machine. We evaluate TPC-H with different scale factors, one and ten. The default scale factor is 10. In the experiment, we exclude TPC-H queries Q8, Q20 and Q21, since each of these queries ran longer than 3 hours.

We run TPC-H benchmark queries on two popular open-source data management systems, PostgreSQL 8.4 and MySQL 5.1. The page size of both systems is set to be 8KB. The buffer size is manually tuned for the best performance on different virtual machines. We examine the monetary cost of evaluating individual queries as well as multiple queries. In the evaluation of multiple queries, we evaluate 30 queries randomly selected from the TPC-H benchmark. Since we mainly focus on the monetary cost incurred by the resource consumption, we exclude installation costs on those systems.

Database Compressions. MySQL supports database compressions, and allows us to investigate database compressions as a tuning of Category B. We use *myisampack* to (de)compress the database. The total storage sizes of tables and their indexes with scale factor of ten are 11.4GB and 17.4GB with and without database compressions, respectively. With compression, the database storage size reduces 35%.

We conducted the experiments on virtual machines offered by Amazon and Rackspace. The operating system is Ubuntu Linux 10.04. The file system is *ext3*. We consider the five virtual machine types in Amazon with different CPU and I/O capabilities. Table 2 summarizes their basic properties listed in Amazon web site [1]. Virtual machine types A1–A5 correspond to *t1.micro*, *m1.small*, *c1.medium*, *m1.large* and *m1.xlarge* in Amazon’s definition. The CPU capability is given in the form of (#core \times #CUPerCore).

We choose the virtual machine types R1–R4 on Rackspace, as shown in Table 3. Rackspace provisions the storage almost proportional to the amount of RAM.

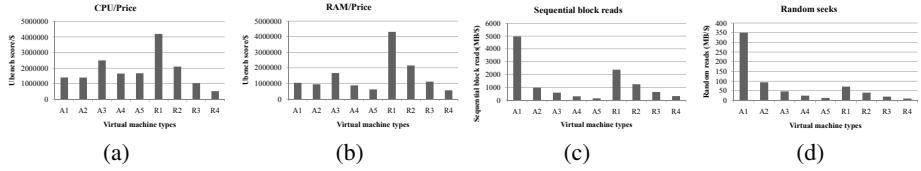


Fig. 2. Monetary efficiency of CPU and memory systems in different virtual machines in Amazon and Rackspace

While the result of a single run is not representative due to the cloud system dynamics, the results of many runs can capture the stability, i.e., forming a band in the measurement [10]. Thus, the average value of multiple runs tends to represent the performance in a long running scenario. In this study, we run each experiment on five virtual machine instances with ten times each, and report the average for the measurements.

4.2 Micro Benchmarks

Figure 2 shows the monetary cost obtained from the micro benchmark results on different virtual machine instances. The block size of IOzone is set to the page size of data management systems (8KB). Comparing the relative monetary cost of different system components among the virtual machines, we find that the relative monetary cost of the CPU is similar to that of RAM; the relative monetary cost of the sequential block accesses is similar to that of random block accesses.

We observe significant differences among different virtual machines. On Rackspace, R1 achieves the lowest monetary cost on all the four micro benchmarks. This results in R1 being the most monetary efficient virtual machine on Rackspace, as we will see in Figure 3. Nevertheless, R1 does not necessarily achieve the lowest monetary cost for all workloads, due to its relatively small main memory capacity as well as storage capacity. On Amazon, there is not a specific virtual machine types dominating the four measured metrics.

4.3 Where Does the Money Go?

Overall, we observed similar monetary cost breakdowns on MySQL to those in PostgreSQL.

Multiple Queries. Figure 3 shows the monetary cost breakdown for running 30 queries in PostgreSQL and MySQL. Since Rackspace does not charge on I/O or storage, the monetary cost includes the execution cost only. In contrast, we observed significant differences in the breakdown among different virtual machines on Amazon. As the virtual machine capability increases, the total execution time decreases, and the amortized storage cost decreases. However, due to the increased price, the execution cost may increase or decrease. As for the I/O cost, as the virtual machine capability increases, the amount of main memory increases and the number of I/O accesses reduces, therefore reducing the I/O access cost. For virtual machines A1–A5, the storage cost is insignificant. The I/O cost and execution costs are two important components in the monetary

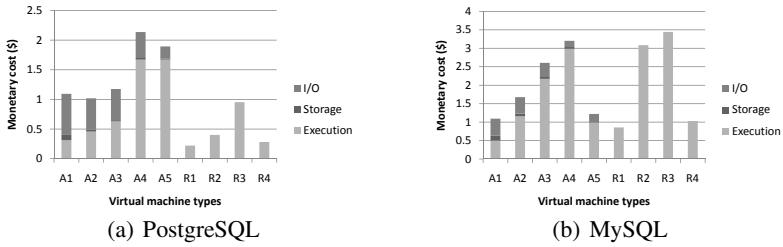


Fig. 3. The monetary cost breakdown of running 30 queries of PostgreSQL and MySQL on Amazon and Rackspace

cost of A1–A5. On the virtual machine with higher prices, the execution cost is more significant and the I/O access cost is less significant.

The time breakdown clearly shows that I/O and execution costs are the most significant components in the monetary cost. We could reduce I/O accesses and/or improve the query processing performance to reduce the monetary cost. They are usually consistent on data management systems, i.e., reducing I/O accesses usually results in performance improvement.

Individual Queries. Figure 4(a) shows the breakdown on monetary cost of PostgreSQL on running each query. Note, we restart the database system before measuring the cost of each query. The monetary cost breakdown of individual queries is similar to those in long-running scenario. The storage cost is insignificant, and the execution cost and the I/O cost are two significant components. Between these two components, the execution cost is the most significant component (more than 50% of the total monetary cost) on Queries Q1, Q6–7, Q12–18 and Q22. The I/O cost is dominated in other queries. Compared with the multi-query scenario, the I/O cost is more significant, because we flush the buffer cache before running individual queries. For other virtual machine types, we also observed similar results.

Different Scale Factors. Figure 4(b) shows the monetary cost breakdown of PostgreSQL running 30 queries with the scale factor one. We observe similar trend as the scale factor of ten. The execution cost is dominated in all virtual machine types, and the

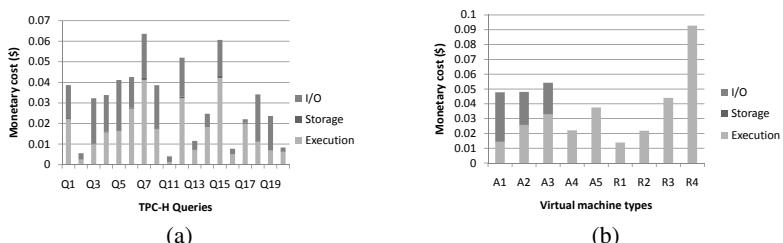


Fig. 4. The monetary cost breakdown: (a) running individual queries on A3; (b) running 30 queries of PostgreSQL with scale factor of one

I/O cost is significant for A1–A3. Since the entire database fits into the main memory of A4 and A5, the I/O access cost is small on those two virtual machine types.

4.4 Performance vs. Monetary Cost

Through a series of experiments on the three categories of tunings, we find that different categories of tuning have significant impact between performance and monetary cost of query evaluations. Category A improves both performance and monetary cost. Category B generally improves both performance and monetary cost with some exceptions, which require special care for the tradeoff between performance and monetary cost. Lastly, Category C usually has conflicts in optimizing performance and optimizing monetary cost. From the micro benchmarks, we have observed that different virtual machines from the same provider or from different cloud providers have different monetary cost on CPU and memory systems. These differences result in the differences in the monetary cost of running TPC-H workloads. Thus, we focus on the results for Categories B and C only.

Tunings in Category B. Compressions. Figure 5 shows the performance and the monetary cost for the simple selection query with sequential scans and TPC-H Q5 with and without compressions on A3. Q5 is a complex five-way join query with sorting and aggregation. Database compressions reduce the I/O cost for all queries. With compression, the execution cost of TPC-H Q5 reduces, and the execution cost for the selection query increases, due to the decompression cost. Overall, database compressions improve monetary efficiency for both queries, but degrade the performance of the selection query.

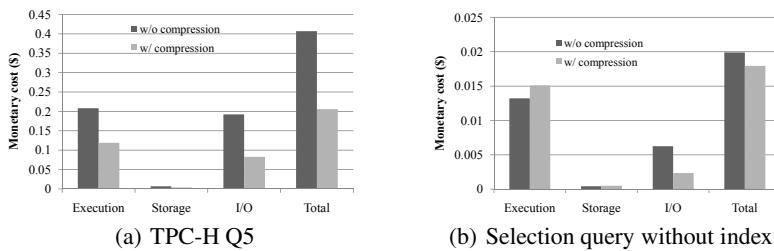


Fig. 5. Monetary costs of evaluating TPC-H Q5 and the selection query with and without compression on A3

Tunings in Category C. Figure 6(a) shows the performance and monetary efficiency of running 30 queries on PostgreSQL and MySQL. Data management systems have been optimized for the performance with various tunings in Categories A and B. The performance and monetary efficiency do not have a clear correlation between each other. The best performing virtual machine type is not the one with the best monetary cost. Moreover, the best performing type varies with different systems.

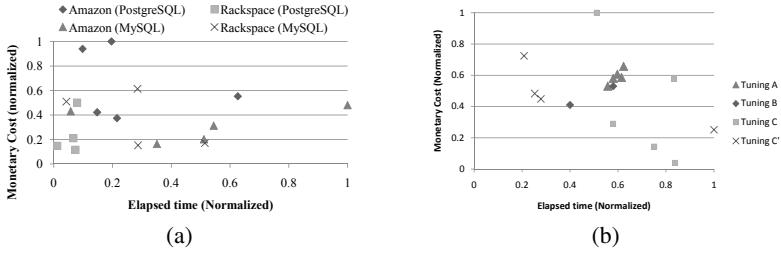


Fig. 6. Performance and monetary cost: (a) on different virtual machines in Amazon and Rackspace, (b) by applying the three categories of tunings individually. Tunings A and B are performed on A3 on Amazon.

Put It All Together. Figure 6(b) illustrates monetary cost and performance (i.e., the elapsed time) for Q5 in TPC-H workload. We have observed similar results in other TPC-H queries. Both monetary cost and elapsed time results are normalized to their corresponding maximum value. Tunings A, B and C(C') belong to Categories A, B and C, respectively. Tuning A is to tune the buffer size within the same virtual machine (i.e., with RAM of 1GB, 2GB, 4GB and 6GB). Tuning B performs Q5 with and without database compressions. Tunings C and C' are to run the query on different virtual machine types offered by Amazon and Rackspace, respectively.

Overall, there is not a clear correlation between performance and monetary cost with all the tunings considered. The point with best performance is not the point with the highest monetary cost, and vice versa. Moreover, we have observed the three cases for the correlation between performance and monetary cost. As a result, the monetary cost varies significantly (the difference is as much as 91% for TPC-H Q5), even with similar performance.

4.5 Impact of Price Structures

Different Charging Methods. Figure 7(a) re-plots the monetary cost of running 30 queries in Amazon with charging according to the RAM hour. We use Rackspace's price, \$0.06 per GB per hour. Clearly, different charging methods affect the monetary cost, since workloads have different consumptions on the resource. With charging on the RAM hour, A3 achieves the best monetary cost. One possible reason is due to the high monetary efficiency of RAM in A3, as shown in our micro benchmarks.

Different Prices among Virtual Machines. On Amazon, the price of execution on A_i is twice as that of $A_{(i-1)}$ ($i \geq 3$). A similar tiering price ratio r exists in Rackspace. Figure 7(b) re-plots of the monetary cost of running 30 queries on A2–A5 in Amazon, varying the ratio, r , in the tiering price. We fixed the price of A5, and vary the ratio of r . As the r increases, the price of A2 decreases. Thus, the monetary cost of A2–A4 increases, and A5 has the same monetary efficiency. The main observation is that the comparison of monetary efficiency among A2–A4 varies with the ratio. For example, A2 has the best monetary cost when $r = 2$, whereas A3 has the best monetary efficiency when $r = 1.5$. Thus, the tiered prices affect the choice on the virtual machine with the best monetary cost.

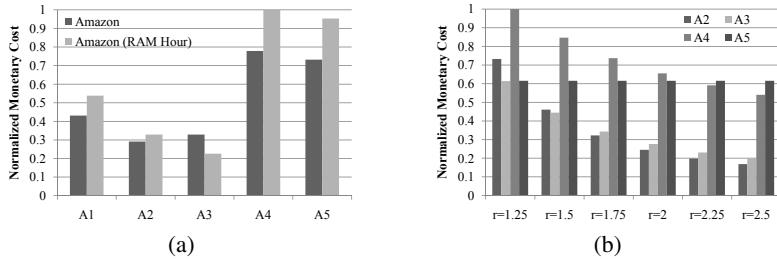


Fig. 7. Normalized monetary cost on A1–A5: (a) with different charging methods, (b) with different tiring price ratios

4.6 Cost Model Evaluations

We observed similar results on PostgreSQL and MySQL, and thus present results for PostgreSQL only. Figure 8(a) shows the monetary cost prediction of PostgreSQL on running each query on A3. The real and estimated monetary costs are normalized to their corresponding maximum values. Our monetary cost model can accurately predict the monetary cost of individual TPC-H queries. Thus, our monetary cost estimation is applicable to different queries.

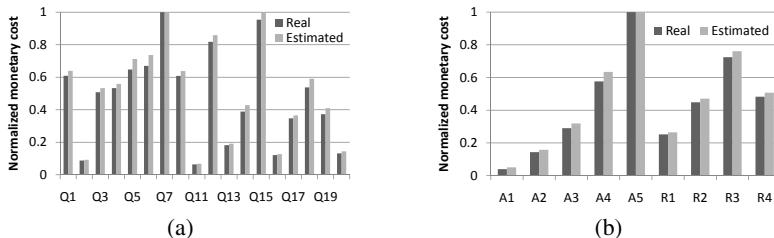


Fig. 8. Monetary cost measurement and estimation: (a) running individual TPC-H queries of PostgreSQL on A3; (b) running TPC-H Q5 of PostgreSQL on Amazon and Rackspace

Figure 8(b) shows the monetary cost prediction of running TPC-H Q5 in PostgreSQL on A1–A5 and R1–R4. Our monetary cost model is able to achieve a good prediction on the trend of the real monetary cost, regardless of virtual machine types.

4.7 Summary

The evaluation of our framework reveals the correlation between performance and monetary cost depends on workloads, price schemes and virtual machine types. We have the following two implications on DBMSs.

First, on the same virtual machine type with fixed prices, the most monetary efficient operating points are usually the best performing for different execution strategies. Our studies confirm that the traditional performance-oriented optimizations continue to

be effective on the same virtual machine. We do see exceptions caused by tunings of Category B on Amazon. One example exception is database compression.

Second, when different virtual machine types and different pricing features are considered, the most monetary efficient operating points are usually *not* the best performing. Our cost model can accurately predict the monetary cost comparison among different virtual machine types.

5 Conclusions

This paper proposes a framework for analyzing the monetary cost of running database systems in the cloud. We evaluate the effectiveness of the framework with database warehousing workloads with two popular open-source DBMSs, PostgreSQL and MySQL, on two cloud providers, Amazon and Rackspace. We find that monetary cost of the same system varies significantly on different virtual machine types and different price definitions. On a specific virtual machine type, the best performing configuration is usually the most monetary efficient. We further develop a monetary cost model with the awareness of virtual machine selections and prices. Our experiments demonstrate that the monetary cost model accurately predicts the monetary cost of query evaluations with different virtual machine offerings and prices.

References

1. <http://aws.amazon.com/ec2/instance-types/>
2. Assunção, M.D., di Costanzo, A., Buyya, R.: Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: HPDC (2009)
3. Coates, J., Gray, J., Nyberg, C.: Performance/price soft and pennysort. Tech. Rep. MSR-TR-98-45, Microsoft Research (1998)
4. Gong, Y., He, B., Zhong, J.: CMPI: Network performance aware MPI in the cloud. IEEE TPDS (2013)
5. Ibrahim, S., He, B., Jin, H.: Towards pay-as-you-consume cloud computing. In: IEEE SCC 2011, pp. 370–377 (2011)
6. IOzone: <http://www.iozone.org/>
7. Kllapi, H., Sitaridi, E., Tsangaris, M.M., Ioannidis, Y.: Schedule optimization for data processing flows on the cloud. In: SIGMOD (2011)
8. Kossmann, D., Kraska, T., Loesing, S.: An evaluation of alternative architectures for transaction processing in the cloud. In: SIGMOD (2010)
9. Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S.: Amazon S3 for science grids: a viable solution? In: DADC (2008)
10. Schad, J., Dittrich, J., Quiane-Ruiz, J.-A.: Runtime measurements in the cloud: Observing, analyzing, and reducing variance. In: PVLDB (2010)
11. Tak, B.C., Urgaonkar, B., Sivasubramaniam, A.: To move or not to move: The economics of cloud computing. In: HotCloud (2011)
12. Ubench: <http://phystech.com/download/ubench.html>
13. Wang, H., Jing, Q., Chen, R., He, B., Qian, Z., Zhou, L.: Distributed systems meet economics: Pricing in the cloud. In: HotCloud (2010)

Efficient Distributed Multi-dimensional Index for Big Data Management^{*}

Xin Zhou, Xiao Zhang ^{**}, Yanhao Wang, Rui Li, and Shan Wang

School of Information, Renmin University of China, Beijing, China
{zhouxin314159, zhangxiao, wyh1990, lrbeckham, swang}@ruc.edu.cn

Abstract. With the advent of the era for big data, demands of various applications equipped with distributed multi-dimensional indexes become increasingly significant and indispensable. To cope with growing demands, numerous researchers demonstrate interests in this domain. Obviously, designing an efficient, scalable and flexible distributed multi-dimensional index has been confronted with new challenges. Therefore, we present a brand-new distributed multi-dimensional index method—EDMI. In detail, EDMI has two layers: the global layer employs K-d tree to partition entire space into many subspaces and the local layer contains a group of Z-order prefix R-trees related to one subspace respectively. Z-order prefix R-Tree (ZPR-tree) is a new variant of R-tree leveraging Z-order prefix to avoid the overlap of MBRs for R-tree nodes with multi-dimensional point data. In addition, ZPR-tree has the equivalent construction speed of Packed R-trees and obtains better query performance than other Packed R-trees and R*-tree. EDMI efficiently supports many kinds of multi-dimensional queries. We experimentally evaluated prototype implementation for EDMI based on HBase. Experimental results reveal that EDMI has better performance on point, range and KNN query than state-of-art indexing techniques based on HBase. Moreover, we verify that Z-order prefix R-Tree gets better overall performance than other R-Tree variants through further experiments. In general, EDMI serves as an efficient, scalable and flexible distributed multi-dimensional index framework.

Keywords: distributed multi-dimensional index, ZPR-tree, big data.

1 Introduction

The ubiquity of location-enabled devices such as intelligent mobiles, GPS and sensors produce tremendous amount of data which are heterogeneous and dynamic in many cases. Naturally, it brings about new challenges for various technologies to manage them efficiently, such as the storage, index, query and analysis of data. In particular, efficient index method is crucial to improve the efficiency of query and analysis.

As is well known, centralized multi-dimensional index technology has been extensively studied in traditional DBMSs. However, they cannot meet the increasingly

^{*} This work is supported by Postgraduate Scientific Research Fund of RUC, No. 13XNH214.

^{**} Corresponding author.

requirements of scalability, high I/O throughput in the context of big data, while NoSQL databases emerge to satisfy relevant needs such as HBase, Cassandra, and MongoDB etc. NoSQL databases have prominent capabilities of high scalability, availability and I/O throughputs. NoSQL databases are generally based on key-value store model which is efficient to support single-key index. And they respond to query requests in millisecond order given one key. Meanwhile numerous applications basically require multi-dimensional query, which NoSQL databases cannot support efficiently. Nowadays, the solutions of NoSQL database to respond multi-dimensional query requests are MapReduce and Table-Scan. But in case of multi-dimensional query request, they are inefficient and costly, especially when the selectivity of query request is low. Consequently, a distributed multi-dimensional index should be appropriate to meet all requirements.

In this paper, we present a distributed multi-dimensional index framework EDMI, Efficient Distributed Multi-dimensional Index in this paper to handle the multi-dimensional query requests for big data management.

In summary, the contributions of this paper are:

- We propose a new Efficient Distributed Multi-dimensional Index—EDMI. It is a new two-layered index framework: the top global index is K-d tree and the bottom index is ZPR-tree.
- We propose a new R-Tree^[12] variant—Z-order Prefix R-tree. ZPR-tree utilizes Z-order prefix to eliminate the overlap of MBRs for R-tree nodes to improve the query performance. Further it makes use of the bottom-up building method based on MapReduce to build ZPR-tree in parallel.
- We present an analysis for the cost of two range query methods of ZPR-tree theoretically.
- We experimentally evaluate EDMI and ZPR-tree in distributed clusters. The results demonstrate that the query performance of ZPR-tree is better than other Packed R-trees and R*-Tree. EDMI has better query performance than other index methods involved in related work.

The rest of this paper is organized as follows. Section 2 presents the related work. We present our new index framework—EDMI in Section 3 in detail, including the architecture and construction of EDMI and the R-tree variants—ZPR-tree. Section 4 discusses query processing on EDMI and analyzes the cost of two range query methods of ZPR-tree theoretically. Section 5 presents an extensive experimental evaluation of EDMI and ZPR-tree. Section 6 concludes our paper and points out our future work.

2 Related Work

Distributed multi-dimensional index for big data management has caught tremendous attention of researchers recently. Several researchers presented great solutions on solving the challenges of distributed multi-dimensional index based on cloud platform. We classify existing index frameworks into two categories according to the distributed storage architecture.

A) The storage architecture of index framework is based on Peer-to-Peer system.

[1] proposes a general index framework of the cloud. The index framework has two layers: global index and local index. Local index is built on local data; while index framework selects nodes from local indexes by respective cost model and organizes them together to form global index. RT-CAN [2] is a specific index framework based on such architecture. In detail, the local index is R-tree, and the network communicate protocol is C² overlay which extends CAN [3] protocol. Another representative framework is QT-Chord [4]. Its local index is MX-CIF quad tree and it is based on Chord overlay.

B) The storage architecture of index framework is based on master-slave style.

EMINC [5] has two-layered indexes as well. R-Tree as the global index is stored in master node, and each slave node indexes local data using K-d tree. A node cube stores the boundary information of each indexed dimension on each slave node. Each leaf node of the R-tree contains a node cube and one or more pointers that point to the slave nodes corresponding to its node cube. MD-HBase [6] leverages a multi-dimensional index structure layered key-value store. Z-order curve is used to partition entire data space into different subspaces, and K-d tree or Quad-Tree is used to group these subspaces which share the longest common Z-value prefix to reduce false positive scans of Z-order curve. The storage of MD-HBase is based on HBase.

Both the storage architectures of index framework have advantages and disadvantages. The major problems of first one are that (1) communication overhead becomes excessive for routing query requests between computer nodes and that network latency severely influences the performance of query;(2) the index framework cannot meet the frequent data insert, delete and update requirement because communication overhead to synchronize the local index and global index is huge. However, accessing time of peer-to-peer system which accesses the local index in local disk is faster than that of the second index framework. The second index framework which adopts master-slave architecture locates the target computer node easily and reduces communication overhead largely. Besides, the fault-tolerant ability of master-slave architecture is stronger than peer-to-peer system and its availability and security is better than peer-to-peer system as well. We adopt master-slave architecture to store our index in the paper correspondingly.

We present a new index framework EDMI to improve the query performance of multi-dimensional query effectively and efficiently. Section 3 will discuss the architecture of EDMI in detail.

3 The Architecture of EDMI

3.1 The Architecture of EDMI

EDMI is a two-layered index framework; Figure 1 presents the architecture of EDMI. The top layer is global index which utilizes point-based adaptive K-d tree [13] to partition entire data space into several subspaces. The bottom layer consists of several ZPR-trees. Each ZPR-tree indexes data items in each subspace. It is worth mentioning

that point-based K-d tree partitions data space into subspaces with the equal amount of data points, which is also efficient for skew datasets. Unlike traditional K-d tree, the split dimension of adaptive K-d tree can be different from each other in the same level. The split dimension of adaptive K-d tree is not cycled. The dimension which corresponds to maximum spread coordinate in the subspace is chosen as the split dimension. And point-based adaptive K-d tree is a complete binary tree, it is effective to code K-d tree using left “0” and right “1” coding to represent the boundary of each subspace.

ZPR-tree is a new R-Tree variant we present in this paper. We will introduce ZPR-tree in detail in section 3.2.

The procedure of constructing EDMI has two phases. The first phase is the construction of K-d tree in the top layer. The second phase is constructing ZPR-trees using MapReduce [7],[8]. The number of R-trees is the same with the number of leaf nodes in K-d tree. In map phase, the task is reading all raw data from HBase and partitioning them into different groups; and in reduce phase, the task is building an ZPR-tree for each group using method in Section 3.3 and writing all ZPR-trees into HBase. Figure 2 illustrates the procedure of construction EDMI using MapReduce.

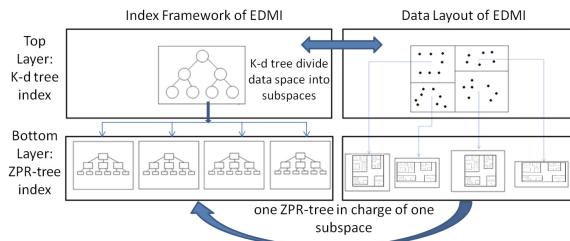


Fig. 1. The architecture of EDMI

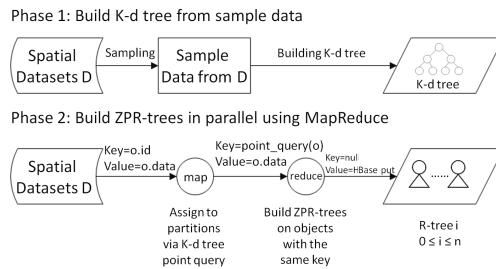


Fig. 2. The procedure of construction EDMI

3.2 Z-Order Prefix R-Tree

Z-Order Curve and Z-Order Prefix Code

Z-order curve is one of the most popular space filling curves, which can be used for linearization of multi-dimensional data. The Z-value of a point in multi-dimensions is

simply calculated by interleaving the binary representations of its coordinate values. Figure 3 illustrates a Z-order curve in 2D space with 6 bits length Z-value. For example, the Z-value of subspace 8 is '001000'. Z-order curve splits data space into 2^n (n is the binary bits of Z-value) number of equal-sized and disjoint subspaces. Z-value is used as the name for each subspace. Z-order curve has two important properties: (1) If subspace A encloses subspace B , the name of subspace A is a prefix of that of subspace B . (2) If subspace A doesn't have the same Z-order prefix with subspace B , the subspace A doesn't intersect with subspace B . We indicate Z-order prefix code of different subspaces in different colors in Figure 3.

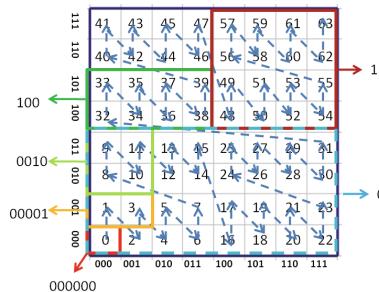


Fig. 3. Z-order Prefix code on Z-order curve

Z-Order Prefix R-Tree

Z-order Prefix R-tree (ZPR-tree) is a new variant of R-tree presented by us. ZPR-tree inherits the properties of Z-order curve. There are some additional new features:

- Z-order curve preserves the locality of data points: all the records in the same R-tree node share the common Z-order prefix according to property (1) above.
- ZPR-tree eliminates the overlap of MBRs for R-tree nodes at the same level according to property (2) of Z-order curve.
- The leaf nodes of ZPR-tree not only store the actual data items and their MBR, but also the common Z-order prefix in the leaf nodes.

Mixing Z-order prefix with R-tree is the prominent point of ZPR-tree. To guarantee the space utilization, entry numbers of one R-tree node is confined between m (where $1 \leq m \leq M/2$) and M . However if data distribution is skewed, there are nodes in ZPR-tree whose entry number is larger than M ; but if we split it into two nodes according to Z-order prefix, entry number of one sub-node may be smaller than m . And ZPR-tree cannot satisfy the fan-out restrictions of R-tree nodes. So we present two policies of ZPR-tree according to various practical demands.

- M-guarantee ZPR-tree. This policy restricts the entry numbers of ZPR-tree node is no larger than M . The advantage of M-guarantee ZPR-tree is that there are no oversized nodes in ZPR-tree and the total MBR area of ZPR-tree nodes is small. However, the nodes number of ZPR-tree is large and the space utilization of ZPR-tree node is so small that I/O frequencies of query request will increase.

- m-guarantee ZPR-tree. This policy restricts the entry numbers of ZPR-tree node is larger than m . m-guarantee ZPR-tree guarantees the space utilization of ZPR-tree node, avoids smaller-sized nodes, and reduces the I/O access times. However, this policy will generate the oversized node if the data distribution is extremely skew. Besides, the total MBR area of ZPR-tree nodes is relatively large and the false positive scans will increase with query processing.

We present the experimental evaluation of these two policies of ZPR-tree in the section 5. And the experimental results prove the theoretical analysis above.

3.3 The Bottom-Up Construction of ZPR-Tree

To improve the insert throughput of ZPR-tree, we adopt bottom-up method to construct ZPR-tree. The bottom-up method is similar to Packed Hilbert R-tree, but takes Z-order prefix into consideration. The basic procedure of this method can be described as follows: (1) Compute Z-values of all data items and sort all data items by their Z-value in ascending order; (2) Partition all data items into different sub-lists according to the M-guarantee or m-guarantee of ZPR-tree strategy, then create one leaf node of ZPR-tree for all items in each sub-list, Z-order prefixes of all items in each sub-list uniquely identify the corresponding leaf node; (3) Generate all internal nodes until the root of ZPR-tree is generated level by level from leaf nodes. In Figure 4, we give the construction procedures of ZPR-tree based on two different split policies from the same list of points. In this example, the capacity of R-tree's node is 5. We present the tree structure of generated ZPR-tree in Figure 5.

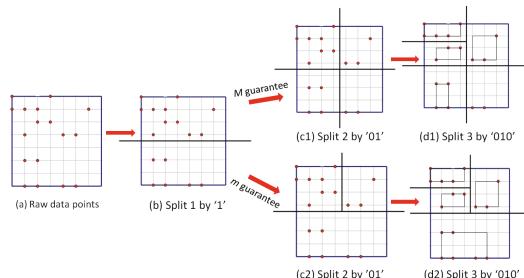


Fig. 4. Bottom-up construction of ZPR-tree using M guarantee and m guarantee

(a)	ZP xl xh yl yh 0 1 5 0 2	ZP xl xh yl yh 100 1 3 4 5	ZP xl xh yl yh 101 0 3 6 7	ZP xl xh yl yh 11 4 6 4 6		ZP xl xh yl yh 00 1 2 0 2	ZP xl xh yl yh 01 4 5 0 0	ZP xl xh yl yh 100 1 3 4 5	ZP xl xh yl yh 101 0 3 6 7	ZP xl xh yl yh 11 4 6 4 6
	(1,0) Z=2 (2,0) Z=4 (1,2) Z=10 (2,2) Z=12 (4,0) Z=16 (5,0) Z=18	(1,4) Z=34 (2,5) Z=37 (3,5) Z=39	(0,6) Z=40 (0,7) Z=41 (1,6) Z=42 (2,6) Z=44 (3,7) Z=47	(4,4) Z=48 (5,4) Z=50 (6,6) Z=60		(1,0) Z=2 (2,0) Z=4 (1,2) Z=10 (2,2) Z=12 (4,0) Z=16 (5,0) Z=18	(4,0) Z=16 (5,0) Z=18 (2,2) Z=12	(1,4) Z=34 (2,5) Z=37 (3,5) Z=39	(0,6) Z=40 (0,7) Z=41 (1,6) Z=42 (2,6) Z=44 (3,7) Z=47	(4,4) Z=48 (5,4) Z=50 (6,6) Z=60

Fig. 5. The tree structure of ZPR-tree, (a) for m-guarantee and (b) for M-guarantee in Figure 4

4 Query Processing on EDMI

4.1 Range Query

Cost Model for ZPR-Tree Range Query

As for processing range query on EDMI, we present two different methods. One is the traditional method for all tree indexes: traverse ZPR-tree from root to leaf (denoted as Traversing-Tree), query all nodes whose MBR intersect with query box and get all results of range query request; another method is called Leaf-Scan with which we scan all leaf nodes whose Z-order prefixes fall into Z-value interval of query box to process range query of ZPR-tree. Leaf-Scan can only be used in ZPR-tree because it is stored in HBase which is based on key-value storage model. We use level number and Z-order prefix as the key in HBase, with format like "0_1100" for a leaf node with "1100" as its Z-order prefix. So nodes of ZPR-tree are stored by their Z-order prefix in dictionary ascend order level by level, which can be scanned easily if given a specific range. In general, traversing tree is more efficient than Leaf-Scan because of less false positive scans; however, we observe that one 'scan' operation is much faster than several 'get' operations to fetch the same number records in HBase. We present a cost model to evaluate the cost of two range query methods and adopt the cheaper one to execute range query.

The main task to evaluate cost of Leaf-Scan is to estimate the ratio of leaf nodes involved in one query box Q . Formula (1) and (2) calculate the numbers of involved leaf nodes corresponding to Q and total cost of Leaf-Scan method.

$$leafs = n \times (z_h - z_l)/z_{max} \quad (1)$$

$$C_{LeafScan} = leafs \times C_{HBaseScan} \quad (2)$$

Where z_l and z_h denote the Z-values of lower bound and upper bound for Q , n denote the total number of leaf nodes in ZPR-tree, z_{max} denote the Z-value interval of ZPR-tree. $C_{HBaseScan}$ is cost of average 'scan' operation of HBase.

Then we present formula (3) and (4) to estimate the cost of Traversing-Tree method. The basic idea which consults^[9] is that the cost of range query is in proportion to the number of disk page accesses which corresponding to 'get' operations in HBase. Formula (3) is used to estimate the number of node access or 'get' operation. Formula (4) is total cost of Traversing-Tree. Formula (5) is the cost of processing range query on ZPR-tree.

$$NA_Total_{(R,q)} = \sum_{l=1}^{h_R-1} \{N_{R,l} \times \sum_{k=1}^d (S_{R,l,k} + q_k)\} \quad (3)$$

$$C_{TraversingTree} = NA_Total_{(R,q)} \times C_{HBaseGet} \quad (4)$$

$$C = \min(C_{LeafScan}, C_{TraversingTree}) \quad (5)$$

In the formula above, d denote the dimension of R-tree, h_R denote the height of R-tree R , $N_{R,l}$ denote number of nodes at level l of R , $S_{R,l,k}$ denote the average extend of node rectangles of R at level l on dimension k , q_k denote extend of query q on dimension k .

We maintain $s_{R,l,k}$ and $N_{R,l}$ as statistics information, store them in the header of R-tree and load them into main memory when R-tree is built for query processing.

Range Query of EDMI

As for processing range query Q on EDMI, firstly query service executes K-d tree range query algorithm for Q to returns all serial numbers of ZPR-trees which intersect with Q . Then query service starts up one range query for each ZPR-tree in parallel. But for each ZPR-tree, it only executes a sub-query which is the intersection of boundary of ZPR-tree and Q . For example, Q is $[0, 20], [0, 30]$ and the boundary of one ZPR-tree r_1 is $[10, 30], [5, 25]$, r_1 only processes range query for $[10, 20], [5, 25]$ instead of $[0, 20], [0, 30]$. After finishing range queries of ZPR-tree, query service combines all results and returns final result set to client. Algorithm 1 elaborates the procedure of handling a range query for area Q .

Algorithm 1. RangeQuery

Input: query Q

Output: Result

```

1: TreeList  $\leftarrow$  KDTreeRangeQuery(KDTree,Q)
2: for each ZPRTree in TreeList do
3:   subQ  $\leftarrow$  FindIntersection(ZPRTree, Q);
4:   if  $C_{TraversingTree} > C_{LeafScan}$  then
5:     Result.add(TraversingTree(ZPRTree,subQ))
6:   else   Result.add(LeafScan(ZPRTree,subQ))
7: end for
8: return Result

```

4.2 Point Query

Point query can be treated as a special case of range query that the size of query box is set to 0. Assume that query service receives a point query request for point P . Firstly query service executes K-d tree point query algorithm for P . Then the serial number of ZPR-tree that may contain P would be picked out. Next query service will execute point query for P on corresponding ZPR-tree. Because ZPR-tree eliminates the overlap of MBRs for R-trees at the same level, a point query at most involves one node at one level. If query service finds point P in ZPR-tree, it will return point P and its information to client, otherwise return null to client.

4.3 K-Nearest Neighbor Query

Given a center C and a parameter K , a nearest neighbor (NN) query requests K data points nearest to the center C , according to some distance metric. In this paper, we use a traditional method for KNN query. First query service sets 1 as the initial radius of query, then executes KNN query on K-d tree and ZPR-tree and gets all results whose distance to C is less than initial radius. If the size of result set is equal with K , query is finished; if the size of result set is larger than K , query service selects top K data

points that are nearest to C as results and finishes the query; if the size of p is smaller than K , expand the radius according to the ratio of the size of result set and K and re-execute the query on new radius, until the size of result set is equal with or larger than K . Algorithm 2 specifies the procedure of KNN query on EDMI.

Algorithm 2. K-nearest neighbor Query

Input: the center C and the parameter K

Output: Result

```

1:   radius  $\leftarrow$  1
2:   do
3:     TreeList  $\leftarrow$  KDTreeKnnQuery(KDTree,C,radius)
4:     for each ZPRTree in TreeList do
5:       Result.add(ZPRTreeKnnQuery(ZPRTree,C,radius))
6:     if Result.size<K then
7:       radius  $\leftarrow$  ExpandRadius(K, Result.size)
8:     while Result.size <k
9:     if Result.size>k then
10:      Sort Result by distance to C in ascend order
11:      Result  $\leftarrow$  Sublist(Result, 0, K)
12:   return Result

```

5 Experimental Evaluation

In this section we present comprehensive evaluations of our prototype which is implemented based on Hadoop 1.0.4 and HBase 0.94.2. Our testing infrastructure includes 8 machines which are connected together. Each computer node has 2 Intel E5645 2.4GHz Xeon CPU, and each CPU has 6 cores. Memory of each computer is 48GB; the disk size is 2TB, running Red Hat Enterprise 5.5 operating system. All the computer nodes are in the same rack, the network is 1Gbps. We perform our experiments a synthetic uniform dataset. This dataset is a 5-dimensional dataset with 500,000,000 points; in this dataset, we simulate 50000 objects' movement on a 2D space, and each object moves 10000 steps. One record in this dataset contains 5 attributes, including ID, timestamp, longitude, latitude, and moving direction.

5.1 Experimental Evaluation on ZPR-Tree

We use the synthetic dataset above to evaluate ZPR-tree and other R-tree variants, and get 1,000,000 points randomly from dataset to construct R-trees. We implement Packed Hilbert R-tree^[11] (PHR-tree), Packed Z-order R-tree (PZR-tree), and R*-tree^[10] to compare with M-guarantee ZPR-tree and m-guarantee ZPR-tree in Section 3. All these tree indexes are implemented in Java and stored in HBase table.

We use average node access times as the metric in point query experiment to evaluate the overlap between MBRs at the same level and the overall area of MBRs. When evaluating the range query of R-trees, total node access (NA) times and total

scanned records (SR) are both important parameters. But in most cases, NA and SR contradict with each other: To reduce NA , we should store more points in one node which increases false positive scans and SR , vice versa. In HBase and other Key-Value stored databases, reducing random reading times is more effective than reducing total reading size in improving the query performance. So in our experiments, we adopt both NA and SR as evaluation criterions of range query performance and give a weighted score that NA (0.6) is more important than SR (0.4). Formula (6) gives how the weighted score is calculated. NA_{R_i} and SR_{R_i} denote NA and SR of R-tree i .

$$WS_{R_i} = 0.6 \times \frac{\left(NA_{R_i} - \min_{j=0}^J (NA_{R_j})\right)}{\min_{n=0}^J (NA_{R_j})} + 0.4 \times \frac{\left(SR_{R_i} - \min_{j=0}^J (SR_{R_j})\right)}{\min_{n=0}^J (SR_{R_j})} \quad (6)$$

Table 1. Construction time of R-trees

time(s)	ZPR-tree m	ZPR-tree M	PZR-tree	PHR-tree	R*-tree
2D	5.684	6.86	6.482	7.627	75.925
3D	5.962	6.625	6.894	7.605	92.302
4D	6.284	7.046	6.962	7.704	98.707
5D	6.487	7.127	7.011	7.805	103.23

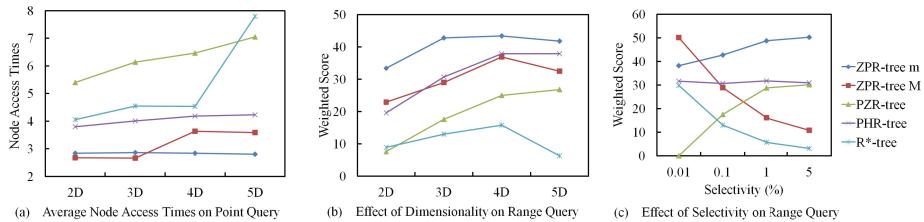


Fig. 6. Experimental results on ZPR-tree and other R-tree Variants

Table 1 depicts the construction time of R-tree variants. We can see that m-guarantee ZPR-tree and Packed Z-order R-tree are fastest on the efficiency of construction; M-guarantee ZPR-tree and Packed Hilbert R-tree are slightly slower because M-guarantee ZPR-tree has more nodes and Hilbert encoding is much slower than Z-order encoding. R*-tree is much slower than Packed R-trees because it cannot support bottom-up construction and spends much time in node split.

Figure 6(a) shows the average node access per point query of R-tree variants. M-guarantee ZPR-tree performs best on 2D and 3D point query because it eliminates the overlap and has smaller overall MBR area. But on 4D and 5D point query, m-guarantee ZPR-tree exceeds M-guarantee ZPR-tree because M-guarantee ZPR-tree's height increases to 4. Packed Z-order R-tree is the worst of all on point query because of much overlap between MBRs and bigger overall MBR area.

Figure 6(b),(c) shows the weighted scores of R-tree variants on range query, (b) for different dimensions under 0.01% selectivity and (c) for different selectivity under the same dimension; the calculation of score has been introduced in Formula (6).

M-guarantee ZPR-tree is the best choice when the selectivity is 0.001%. But as selectivity increases, m-guarantee becomes the better one because its *NA* is much less than other R-tree variants. And M-guarantee ZPR-tree always has the least *SR* of all. R*-tree becomes weak quickly when the dimension reaches 5. The experimental results reveal that M-guarantee ZPR-tree performs best of all on point query and low selectivity range query, m-guarantee ZPR-tree performs best of all on high selectivity range query and ranks the second on point and low selectivity range query, thus making it the best overall performance of all R-tree variants above.

5.2 Experimental Evaluation on EDMI

We complete a series of experiments to compare the performance of EDMI with MD-HBase for three storage models (Table per Bucket, Region per Bucket and Share Table). We test point, range and KNN query on the synthetic dataset above. We use the average latency time of query processing to evaluate query performances.

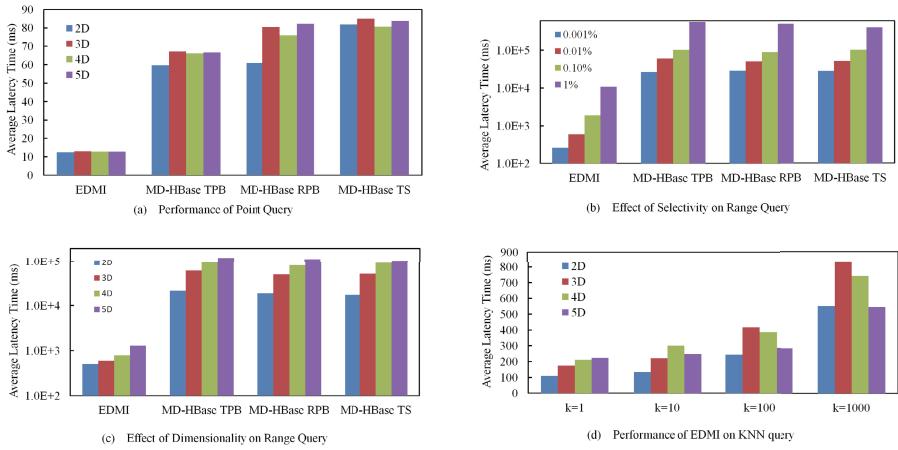


Fig. 7. Experimental results on EDMI and MD-HBase

Figure 7(a) illustrates the performance of EDMI and MD-HBase on point query. The result shows that both indexes respond a point query request in less than 100 milliseconds but EDMI is about 5 times faster than MD-HBase. Figure 7(b) and 7(c) shows the effect of selectivity and dimensionality on the range query of EDMI and MD-HBase, (c) for different selectivity on the same dimension and (d) for different dimensions under 0.01% selectivity. We can conclude that EDMI exceeds MD-HBase over 10 times on range query and the gap becomes larger as selectivity and dimension increase. This is mainly because EDMI uses ZPR-tree in the bottom layer, but MD-HBase employ scan. Figure 7(d) shows the performance of EDMI's KNN query, which turns out that EDMI can process KNN query efficiently on different dimensions and *K* parameters. In general, EDMI is much more efficient than MD-HBase on point and range query processing, and it can process KNN query efficiently as well.

6 Conclusion

In this paper, we propose a new multi-dimension index framework—EDMI. EDMI is based on HBase and it has two layers: the top layer is K-d tree and the bottom layer is a group of ZPR-trees. ZPR-tree is a new variant of R-tree using Z-order prefix to avoid the overlap of MBRs for R-trees on multi-dimensional point data and it has the equivalent construction speed of Packed R-trees and better query performance than other Packed R-trees and R*-tree; and we verify this in our experiment by comparing ZPR-tree with other R-tree variants. EDMI efficiently supports point, range and KNN query on HBase. In our experiments, we compare EDMI with MD-HBase and prove that EDMI has much better multi-dimensional query performance.

We only focus on efficient construction from static datasets and high multi-dimensional query performance of EDMI now, and then we will pay more attention on supporting frequent insertion and update operations on EDMI in the future. Moreover, ZPR-tree will be extended to index various types of data not merely point data.

References

1. Wu, S., Wu, K.-L.: An indexing framework for efficient retrieval on the cloud. *IEEE Data Eng. Bull.*, 75–82 (2009)
2. Wang, J., Wu, S., Gao, H., Li, J., Ooi, B.C.: Indexing multi-dimensional data in a cloud system. In: *SIGMOD Conference 2010*, pp. 591–602 (2010)
3. Ratnasamy, S., Francis, P., Handley, M., Karp, R.M., Shenker, S.: A scalable content-addressable network. In: *SIGCOMM 2001*, pp. 161–172 (2001)
4. Ding, L., Qiao, B., Wang, G., Chen, C.: An efficient quad-tree based index structure for cloud data management. In: Wang, H., Li, S., Oyama, S., Hu, X., Qian, T. (eds.) *WAIM 2011. LNCS*, vol. 6897, pp. 238–250. Springer, Heidelberg (2011)
5. Zhang, X., Ai, J., Wang, Z., Lu, J., Meng, X.: An efficient multi-dimensional index for cloud data management. In: *CloudDB 2009*, pp. 17–24 (2009)
6. Nishimura, S., Das, S., Agrawal, D., Abbadi, A.E.: MD-HBase: A scalable multi-dimensional data infrastructure for location aware services. In: *Mobile Data Management (1) 2011*, pp. 7–16 (2011)
7. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation*, vol. 6, p. 10 (December 2004)
8. Cary, A., Sun, Z., Hristidis, V., Rishe, N.: Experiences on Processing Spatial Data with MapReduce. In: Winslett, M. (ed.) *SSDBM 2009. LNCS*, vol. 5566, pp. 302–319. Springer, Heidelberg (2009)
9. Theodoridis, Y., Stefanakis, E., Sellis, T.: Efficient Cost Models for Spatial Queries using R-trees. *IEEE Transactions on Knowledge and Data Engineering* 12(1) (2000)
10. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: *ACM SIGMOD Conf.*, pp. 322–331 (1990)
11. Kamel, I., Faloutsos, C.: Parallel R-Trees. In: *Proc. of ACM SIGMOD Conf.*, pp. 195–204 (1992)
12. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: *ACM SIGMOD Conf.*, pp. 47–57 (1984)
13. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM TOMS* 3(3), 209–226 (1977)

Fast Multi-fields Query Processing in Bigtable Based Cloud Systems

Haiping Wang, Xiang Ci, and Xiaofeng Meng

School of Information, Renmin University, Beijing, China
`{wanghaiping1022,cixiang,xfmeng}@ruc.edu.cn`

Abstract. With the rapid increase of data sizes, enterprise applications are migrating their backend data management and analytic systems into cloud based data management systems. Bigtable is among one of the major data models used by cloud storage systems as their storage layer. Such systems provide high scalability and schema flexibility, and support efficient point and range based queries based on rowkeys. However, Bigtable based systems have limited support on non-rowkey based queries and multiple-fields based queries, due to much overhead on invoking extra scanning of data. In this paper, we develop a system *TNBGR*(Telecom Network Browsing Gateway Records) on managing and querying large scale telecommunication data. TNBGR is built on top of HBase and MapReduce, with a focus on optimizing multi-fields query processing. TNBGR provides a novel application and system resource aware data allocation strategy to minimize data access through multi-layer region partitioning, resource parameterization, and balanced region distribution. The query composition dynamically updates application parameters based on tracked system statistics and automatically translates queries for MapReduce. Through additional query optimization by improving region locality, TNBGR achieves high efficiency on supporting multi-field queries. The experimental results show that our solution improves the performance of the queries by about 5 and 18 times respectively.

Keywords: HBase, Bigtable, Multiple-Fields Query Processing.

1 Introduction

Enterprise companies traditionally rely on RDBMS or parallel DBMS based solutions to manage and query large datasets in their business applications. For example, telecommunication companies in China keep users' CDR(Called Detailed Records) data in data management systems. To facilitate illustration, in the rest of this paper, we use a table named R to represent the schema of the CDR data. The schema of table R is simplified as:

$$R(msisdn, url, ts, size, otherdata)$$

Here $msisdn$ is the user's phone number. url is the target web site and ts is a timestamp when the user start this accession. $size$ records the size of data

traffic. All the other related data is stored in column *otherdata*. Based on the government's regulations, telecom companies only need to store latest access history data within a period of time, i.e., 90 days. Many applications can be built based on these CDR data. Following are two typical scenarios:

Q1: *For a given user, find the top N web sites the user has accessed and the corresponding network traffic during a period of time.*

Q2: *Return top N users who had accessed a web site during a period of time and the aggregated network traffic for each user.*

In previous solution, all the *CDR* data was stored in a distributed relational database and queries are executed in SQL. However, nowdays more and more users surf the Internet via their smart phones instead of their computers. The scale of access log data is growing rapidly into 100TB level and still keeps on increasing. Enterprise companies start to migrate their background data into cloud data management systems such as HBase[5] or others and use MapReduce[2] to accelerate query processing. Some of such systems use Bigtable[4] data model in their underlying storage layer, referred as *Bigtable based cloud systems* in this paper.

Bigtable based cloud systems such as HBase support row key based point query and range query efficiently. However, for queries based on multiple fields or on non-rowkey field, many redundant records have to be scanned, which could lead to unsatisfactory performance. In this paper, we present a cloud based data management and query system *TNBGR*(Telecom Network Browsing Gate way Records). We build TNBGR by extending and optimizing from HBase and MapReduce [2], to provide highly efficient query support of typical multiple-fields queries for telecom applications.

The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 presents our data allocation strategy and implementation approaches. Query processing is discussed in Section 4, including the query processing workflow, and query decomposition. Section 5 discusses additional query optimization techniques. Performance study is discussed in Section 6, followed by conclusion.

2 Related Work

2.1 Query Optimization through MapReduce

MapReduce is a software framework introduced by Google to support distributed computing on large data sets on clusters of computers. Computational processing can occur on data stored either in a unstructured filesystem or in a structured database. It has two steps named *map* and *reduce* and allows for distributed processing of the map and reduction operations. To accelerate multiple-fields query in Bigtable based cloud systems like HBase, some previous effort was made by using MapReduce job to do parallel query processing.

When MapReduce job is used to do multiple-fields query processing on Bigtable based cloud systems, each *maptask* does sub query on one tablet and query

parallelism is achieved through MapReduce. MapReduce tasks can fetch data directly from the distributed file system or from the database layer, but redundant data scan can not be avoided.

2.2 Query Optimization through Indexes

Another way to improve the efficiency of multiple-fields query on Bigtable based cloud systems is through using indexes. According to the data structures, we can divided the indexes into two categories: one-dimensional indexes and multiple-dimensional indexes.

ITHBase[6], IHBase[9], CCIIndex[11] and Asynchronous views[1] are four representative related work that use several one-dimensional indexes to accelerate multiple-fields query processing. For each column that is frequently used by user queries, a one-dimensional index was build on it.

RT-CAN[8], QT-Chord[3], EMINC[10] and A-Tree[7] are four types of multi-polygonal indexes suitable for cloud data management systems. RT-CAN integrates CAN based routing protocol and the R-tree based indexing scheme to support efficient multi-dimensional query processing in a Cloud system. QT-Chord integrates Quad trees and Chord protocol together. EMINC[10] uses K-d tree as its local index and R-Tree as its global index. A-Tree[7] is one types of R-tree with bloom filter. The upper four indexes can support multiple-fields query very well with good scalability. But the size of index data should be very large, and the cost of maintaining and updating these indexes is very high too.

3 Data Allocation Strategy

One approach to accelerate the efficiency of a query is to reduce the candidate data that will be accessed during the query procedure. In this paper, we try to reduce the candidate data during the query. To achieve this, we propose a novel data allocation strategy named MDRO algorithm (Multiple-Dimensional Region Organization Algorithm). MDRO algorithm consists of three parts: fields selection, region organization strategy and row key generation algorithm.

3.1 Fields Selection

Queries based on the row key are normally efficient in HBase. To provide efficient queries, we should try to transform our queries into queries based on the row key. As we mentioned above ,there are two typical queries: **Q1**and **Q2**, so the query constraints are based on three fields: user's phone number *msisdn*, the *url* of the web site and access timestamp *ts*. Thus we would like to use *msisdn*, *url* and *ts* for row key generation.

However, not all fields related with query constraints are suitable for row key generation. We generalize three rules to classify fields based on suitability for used as row key, discussed below.

- *Rule 1: Identification.* The selected fields should uniquely identify a row key. This rule is due to the uniqueness requirement of a row key.
- *Rule 2: Usefulness for queries.* This is because in Bigtable based cloud systems the efficiency of query based on row key is much better than that on non row key. So we should select as many fields as possible for those which are frequently used with query constraints.
- *Rule 3: Conducive to data and access workload distribution.* This rule is based on the consideration of system workload balance.

Based on the upper three rules, we chose fields *msisdn* and *ts* for row key generation in our solution for CDR data storage.

3.2 Region Organization

In HBase, each big table is partitioned into a lot of regions. Most data maintenance and management operations are based on regions. The organization of regions affects the efficiency of data insert, delete, update and queries. We propose to organize the regions into a multiple layer grid tree (MLGT). The architecture of MLGT is shown in Fig. 1:

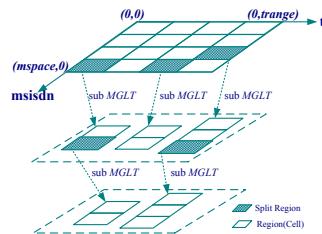


Fig. 1. Multiple Layer Grid Tree(MLGT)

In *MLGT*, regions are firstly organized into a two dimensional grid with two axes: *ts* as the horizontal axis and *msisdn* as the vertical axis. The domain of *ts* is initially partitioned into N parts while *msisdn* is partitioned into M sub ranges. Each cell in *MLGT* maps to a region. As data is inserted into the table, the size of a region may reach the threshold, then the corresponding region will be split into a sub grid. By changing the parameter M and N , we can minimize region splits to reduce the depth of *MLGT*.

In our solution, for each sub grid in *MLGT*, the value of N is always equal to 1 (the reason will be explained later in discussions on row key generation algorithm). We always count the work load of data insert operations for each region, and the value of parameter M is decided by the work load statistics of the given region. The data structure of *MLGT* can be viewed as follows:

```

TypeDef struct MLGT{
    int N;
    int M;
    bool bm[m][n];
    long inserts[m][n];
    long trange;
    long mspace;
    Map <RegionID, MLGT > SR;
} MLGT

```

The variables are described below:

- *N*: *int* type, which represents the number of splits happened for the grid’s total time range, and the granularity of time is millisecond.
- *M*: *int* type, which represents the number of splits for the grid’s total *msisdn* space.
- *bm*[*m*][*n*]: *boolean* type, which represents a two-dimensional array, with size of *M* * *N*. if *b*[*m*][*n*] = *true*, it means that the region in the *m*th row and *n*th column of the given *MLGT* has been split and has a key/vlaue pair in map *SR*. The key of the key/value pair is the *RegionID* and the value is the root node of its sub multiple layer grid tree.
- *insert*[*m*][*n*]: *long* type, which represents a two-dimensional array, with size of *M* * *N*. *insert*[*m*][*n*] stores the statistics of insert workload for the corresponding region.
- *trange*: *long* type, which represents the total number of possible *ts* for the given multiple layer grid tree.
- *mspace*: *long* type, which represents the total number of possible *msisdn* for the given multiple layer grid tree.
- *SR*: a map, which maps a region which has been split before to the root node of its child multiple layer grid tree.

In our solution, *RegionID* is the *startkey* of a given region. Thus when a region is split, the *RegionID* of the parent region keeps the same to make it convenient to track split regions.

Now we will discuss the value of *M* and *N*. For the root layer of each table’s *MLGT*, the initial value of *M* and *N* are affected by the following eight parameters:

- *Sizeof(R)*: an estimated value of the size of the table *R*.
- *NodeNumber*: the number of nodes of the cluster.
- *RegionTruncateTime*: the time for the cluster to drop and recreate a region.
- *SystemBearableTime*: the time duration that the upper applications can bear for the cluster to be out of service.
- *ConcurrencyDegree*: the number of regions that each node can concurrently accesses.
- *RegionSize*: the size of a region.
- *Domain(ts)*: the interval threshold of the *ts* column.
- *Domain(msisdn)*: the interval threshold of the *msisdn* column.

Suppose the replication factor of the underlying Hadoop distributed file system is 3. Then the value of M and N are initialized by following five rules:

Rule 1: The value of $M \times N$ should be close to the cluster's finally region number for table R .

$$M \times N \dashrightarrow \frac{\text{Sizeof}(R)}{\text{Regionsize}} \quad (1)$$

Rule 2: Region truncating time has to be less than system bearable down time.

$$\text{RegionTruncateTime} \times M < \text{SystemBearableTime} \quad (2)$$

Rule 3: The system node number multiplying the degree of concurrency should be large than the number of splits for the grid's total $msisdn$ space.

$$\text{NodeNumber} \times \text{ConcurrencyDegree} > M \quad (3)$$

Rule 4: $\text{Domain}(ts)$ must be evenly divisible by N .

$$\text{Domain}(ts) \% N == 0 \quad (4)$$

Rule 5: $\text{Domain}(msisdn)$ must be evenly divisible by M

$$\text{Domain}(msisdn) \% M == 0 \quad (5)$$

3.3 Row Key Generation

In this section, we will discuss the details on how to generate a row key suitable for the region organization algorithm with a given $msisdn$ and ts that are selected out by the fields selection rules. In this paper, to maximize the system's performance, we design the row key generation algorithm by following constraints below:

- The row key must be unique.
- The row key generation function should be conducive to data distribution.
- Given a $msisdn$, the row key function is a time-based continuous function, which can make $Q1$ more efficient.
- To accelerate $Q2$, we should try to support range query without $msisdn$.
- When a region is split, the row key of all the records in the region keeps the same.

Based on the above constraints, we propose our row key generation algorithm. The row key of a given CDR record is decided by three parameters: $msisdn$, ts and the $MLGT$ of its corresponding table. Equation 6 is the row key generation function:

$$\text{rowkey} = \text{key}(msisdn, ts, MLGT) \quad (6)$$

Algorithm 1 is the actual body of the row key function, and each row key consists of two parts: $basekey$ and $offset$. The $basekey$ is the $startkey$ of its corresponding region while $offset$ is the offset. The value of $basekey$ can be calculated by function $\text{Base}(m, n, tr, mr, N)$ while function $\text{Offset}(m', n', tr)$ is used to generate the value of $offset$.

$$\text{Base}(m, n, tr, mr, N) = (m * N + n) * tr * mr \quad (7)$$

$$\text{Offset}(m', n', tr) = m' * tr + n' \quad (8)$$

Algorithm 1: row key generation algorithm

Input: $msisdn, ts, MLGT$
Output: $rowkey$

Variables:
 $MLGT$: the root multiple layer grid tree
 $msisdn$: user's phone number
 ts : timestamp

Procedure:

```

1:    $rowkey, basekey, tmpkey, offset = 0;$ 
2:    $mr, tr, m, n = 0; \quad tmpts = ts;$ 
3:    $tmpmsisdn = msisdn; \quad flag = false;$ 
4:   do
5:      $mr = MLGT.mspace / MLGT.M;$ 
6:      $tr = MLGT.trange / MLGT.N;$ 
7:      $m = tmpts % mr;$ 
8:      $n = tmpts / tr;$ 
9:      $tmpkey = Base(m, n, tr, mr, MLGT.N);$ 
10:     $basekey = basekey + tmpkey;$ 
11:     $tmpmsisdn = tmpmsisdn \% mr;$ 
12:     $tmpts = tmpts \% tr;$ 
13:    if ( $MLGT.bm[m][n]$ )
14:       $flag = true;$ 
15:     $MLGT = MLGT.SR.get(basekey);$ 
16:    else  $flag = false;$ 
17:  while ( $flag$ )
18:   $offset = \text{Offset}(tmpmsisdn, tmpts, tr);$ 
19:   $rowkey = basekey + offset;$ 
20: Return  $rowkey;$ 
End Procedure:

```

Note that the row key generated by algorithm 1 can not comply with constraint 5. To solve this contradiction, we set the value of N for each $MLGT$ equal to 1 except that the $MLGT$ is the root node of the given table's multiple layer grid tree. This additional parameter setting can help us make sure the row key of all the records in a region keep the same even if when the region got split.

4 Query Processing

4.1 Query Work Flow

As showed in Fig. 2, the query work flow consists of two parts: the query decomposition component and the MapReduce component. The query decomposition component is response for generating the query plan, while the MapReduce component translates the query plan to MapReduce jobs to execute the query.

There are five steps to finish a query submitted by a client:

Step 1: The client submits a multiple-fields query request.

Step 2: The query decomposition component will set parameters, including parameters for tables and meta info for regions. The setting parameters include the table name stored in HBase, and parameters for the $MLGT$ of the given table.

Step 3: The query decomposition component sends these MapReduce job parameters to the MapReduce component. With these parameters, the MapReduce component creates a MapReduce job and submits it to the jobtracker.

Step 4: The fourth step executes the MapReduce job, where each map task accesses one candidate region.

Step 5: This step returns the final query results to the client.

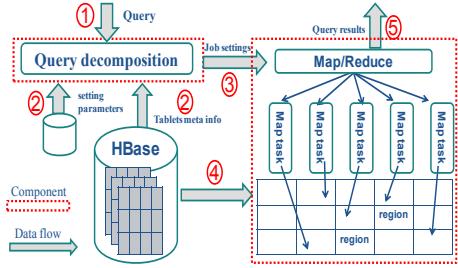


Fig. 2. Query work flow

4.2 Query Decomposition

Once the query decomposition component receives a query request, the first thing it does is to parse the query statement to collect the related fields and constraints. Then it will classify the fields into two sets: *RFQ*(Row key related Fields in the Query) and *NRFQ*(Non Row key related Fields in the Query). At the same time, it also divides the query constraints into two categories: constraints on fields that are elements in *RFQ* and constraints on fields that are elements in *NRFQ*. It finally determines the candidate regions in the query table's multiple layer grid tree.

To facilitate discussion, we use arabic numerals to identify the cells of table R' 's *MLGT*. As showed in Fig. 3, region 12 has been split into 3 regions: 21, 22, 23. Region 13 has been split into region 17 and 18 while region 15 is split into region 19 and 20. Note that these arabic numbers are not the *regionID* of its corresponding region: the *regionID* is the start key.

Before discussing the details of the query decomposition procedure, we will introduce another data structure named *RegionInputSplit* used for packaging the query parameters for each MapTask, which stores the *regioninfo*, the candidate *startRow* and *endRow* of the corresponding region. The range $[startRow, endRow]$ is a subset of the region's key range. The candidate number of *RegionInputSplit* for each query, the *regioninfo* and the range $[startRow, endRow]$ of each *RegionInputSplit* are decided by query constraints based on *RFQ* fields. Other query constraints based on fields which are elements of *NRFQ* are packaged as filter instance and used for the *map()* function to do additional filtering operation if necessary. The data structure of *RegionInputSplit* can be viewed as follows:

```
Typedef struct RegionInputSplit{
    Filter filter;
    HRegionInfo regioninfo;
    key startRow;
    key endRow;
}RegionInputSplit
```

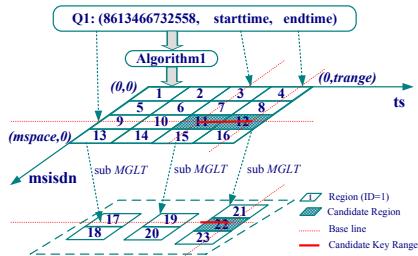


Fig. 3. Query decomposition for Q1

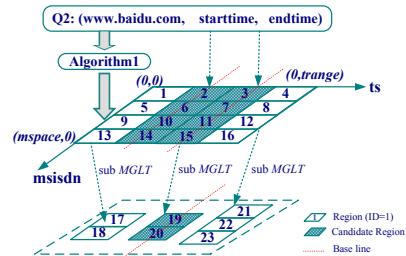


Fig. 4. Query decomposition for Q2

For Q_1 , as showed in Fig. 3, RFQ is $\{msisdn, ts\}$, and $NRFQ$ is an empty set. There are three query constraints on field ts and $msisdn$: $starttime < ts < endtime$ and $msisdn = 8613466732558$; With the two query constraints on field ts we can know that the candidate regions are cells in the third and fourth columns of the grid (marked with imaginary red line). What's more, based on the query constraints on $msisdn$ field, we can derive that the candidate regions for Q_1 are region 11 and 12, since region 12 has been split into region 21, 22, 23. With the child multiple layer grid tree of region 12 and the given $msisdn(8613466732558)$, the final candidate regions for Q_1 are region 11 and 22. The detailed procedure of query decomposition for Q_1 can be seen in Fig. 3.

Since all the query constraints of Q_1 are based on the fields in RFQ , for each record delivered from the record reader during the map phase, if it matches the query constraints, the $map()$ function does not need to perform any filtering operation and only needs to perform a region level aggregation operation.

For Q_2 showed in Fig4, RFQ is $\{ts\}$, and $NRFQ$ is $\{url\}$. There are two query constraints on field ts . The first is $starttime < ts < endtime$. This can help us to find out the candidate regions of query Q_2 are region 2, 3, 6, 7, 10, 11, 14, 15. Here region 15 has already been split into two sub-regions: region 19 and region 20. For the query decomposition component to locate the candidate regions without query constraint on $msisdn$, we need to access all the tuples of each candidate region, i.e., tuples in the region's key range $[startKey, endKey]$ have equal query range $[startRow, endRow]$ of each $RegionInputSplit$. The constraint $url = "www.baidu.com"$ will be packaged as a filter instance into each $RegionInputSplit$. The $map()$ function in the map phase of the MapReduce is responsible for verifying whether the record matches the query constraint $url = "www.baidu.com"$. The procedure is expressed in Fig. 4.

For both query Q_1 and Q_2 , after selecting out all the candidate regions, the query decomposition component will query the root region with the given table name to get the meta info of these candidate regions such as the location of each region(the region server that manages the region).

Finally the query decomposition component will generate a query plan and transform the query plan into a MapReduce job, where each map task accesses one candidate region and performs a region-level aggregation and shuffles the result to the reducer.

5 Query Optimization

5.1 Region Localization

Since the network transmission overhead affects the query efficiency, one way to reduce the query time is to reduce the network overhead: the RPC (Remote Procedure Call Protocol) time between datanodes and region servers and the RPC time between region servers and tasktrackers.

In $TNBGR$, we modify the region allocation algorithm in HBase to try to assign as many regions as possible in the nodes which hold most of the corresponding chunks. First, we set the region size multiple times bigger than the size

of each chunk in the underlying HDFS system. Then we assign the region to the physical node which holds most replicas of its corresponding chunks periodically.

5.2 Parallel Aggregation

Since both $Q1$ and $Q2$ are aggregate queries, the filter operations for different records are independent with each other. In TNBGR, multiple threads are introduced to aggregate the records in parallel. For each *maptask*, one single *reader thread* is introduced to read all the records from HDFS and has them cached in the buffer at first. Then multiple *query threads* filter the records in parallel, with each thread accessing one segment. Finally the *main thread* of the *maptask* collects all the results from *query threads* and performs a region level aggregation and shuffles the region level aggregation result into the reduce side.

6 Experiment Study

We present a detailed evaluation of our solution. We implement our system using HBase 0.90.2 and Hadoop 0.20.2. We evaluate the trade-offs associated with the different implementations for the storage layer and compare our solution with the base line solution. In the base line solution, the row key also consists of ts and $msisdn$, but the row key generation function is very simple:

$$\text{key}(msisdn, ts) = ts.toString() + msisdn.toString() \quad (9)$$

The row keys are sorted in alphabetical order and regions are organized into a linear row key space set by default by HBase and have not been pre-split. A region is split into two child regions in the middle of its row key space when its size reaches the region size threshold.

To finish $Q1$, for each possible ts value in range $(starttime, endtime)$, with the given value of $msisdn$ and the row key generation algorithm as showed in equation 9, one target row key is calculated. These discrete target row keys are then packaged into different *inputsplits* for MapReduce jobs according to the regions they belong to. Query $Q1$ can be finished without any unnecessary data scan. But for $Q2$, a lot of unnecessary data needs to be scanned, since for each t , all the records with ts equal to t will be accessed.

6.1 Experiment Setup

Our testing infrastructure includes eight machines which are connected together to simulate cloud computing platforms. One node is used as a master node and the other seven nodes are used as slave nodes. Each node contains two Intel Quad-Core 2.4GHz CPU, 16GB of main memory and 7 TB hard disk. The OS is Ubuntu 10.10, and the network communication bandwidth is 1Gbps.

The experiment data set is telecom CDR(Called Detail Record) data and the schema of table R is

$$R(msisdn, url, ts, size, otherdata)$$

During the experimental phase, R has two column families named FD (frequently accessed data) and OD (other data). $msisdn$, url , ts , $size$ are four qualifiers of FD in HBase. The size of table R is 1 TB with 2 billion records, and each record has 512 bytes. The $msisdn$ is a long key with value range of [8610000000000, 861999999999] and ts is a randomly generated long value with the value in range of [0, 90 * 24 * 3600 * 1000]. The value of url for each tuple in zipfian distribution. The query time duration is the first two days of the 90 days. The size of each region is set to 512MB. The parameters $mapred.tasktracker.map.tasks.maximum$ and $mapred.tasktracker.reduce.tasks.maximum$ are both set to 7. We conduct two types of experiments:

- Performance evaluation experiment. Compare the query time of $Q1$ and $Q2$ with the baseline solution, and the query time based on our data allocation strategy before optimization and after optimization.
- Performance on impact parameters. By changing the initial value of M and N , we evaluate the query time of $Q1$ and $Q2$ by tuning the settings of the initial value of M and N .

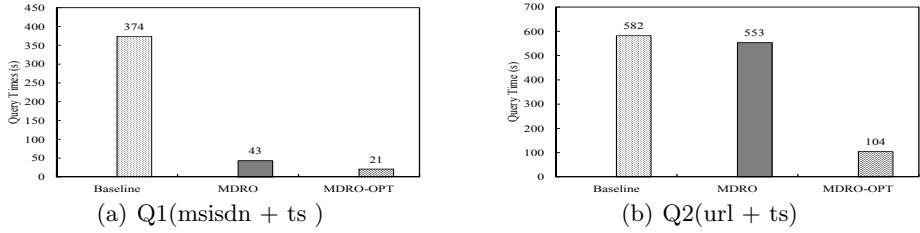
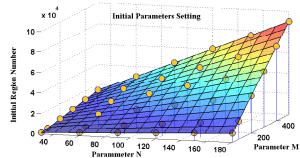


Fig. 5. Performance evaluation result

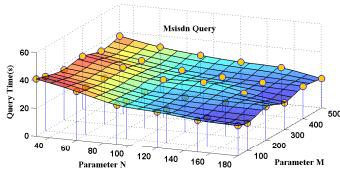
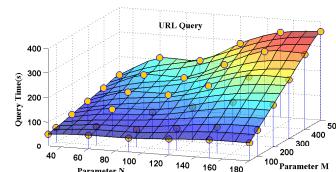
In our performance evaluation experiment, the initial value of M for the root layer grid of table R 's and $MLGT$ is set to 200 and N is set to 45, thus the total number of regions of table R is 9000. For query $Q1$, the number of candidate regions is 2, and for query $Q2$, the number of candidate regions is 400. The experimental results demonstrated in Fig. 5 show significant performance improvement for both queries with our solution. For $Q2$, the query time is reduced to about 20% of the query time from the baseline solution, and for $Q1$, the query time is reduced to 5.6% of the query time from the baseline solution. For $Q2$, our query optimization techniques provide major improvement of the performance.

Fig. 6 shows the settings of impact parameters used in the experiment study. The initial value of N for the first layer of R 's $MLGT$ is scaled up from 30 to 180 while M consist of values of {50, 100, 200, 300, 400, 500}. The query constraints are the same as the performance evaluate experiment. After query composition, the number of candidate regions (the number of map tasks for the corresponding MapReduce job of the query) can be seen in Table 1. The term MR means the candidate regions for query $Q1$ ($msisdn$ based) and UR means candidate regions for query $Q2$ (url based). Fig. 7 and 8 show the query time of $Q1$ and $Q2$.

**Fig. 6.** Initial Parameter Settings**Table 1.** Candidate Regions

<i>N</i>	30	60	90	120	150	180
<i>M</i>	MR UR	MR UR	MR UR	MR UR	MR UR	MR UR
50	1 50	2 100	2 100	3 150	4 200	4 200
100	1 100	2 200	2 200	3 300	4 400	4 400
200	1 200	2 400	2 400	3 600	4 800	4 800
300	1 300	2 600	2 600	3 900	4 1200	4 1200
400	1 400	2 800	2 800	3 1200	4 1600	4 1600
500	1 500	2 1000	2 1000	3 1500	4 2000	4 2000

From Table 1 and Fig. 7, we discover that the query time of Q_1 is mainly influenced by the value of N : as the value of N increases, the query time decreases. This is because as the value of N increases, the key range for each candidate region decreases, and it reduces the execution time of each map task. Although the parameter N influences the number of map tasks of the corresponding job, the number of map tasks does not reach the threshold of map slots that can be executed simultaneously in the cluster(49 in our cluster). We can conclude that once the value of N reaches the threshold, the query time will increase as well.

**Fig. 7.** Msisdn Query**Fig. 8.** URL Query

From Table 1 and Fig. 8, we find that the query time of Q_2 are influenced by the values of both N and M : as the values of M and N increase, the query time increases too. As the values of N and M increase, although the size of each region deceases and reduces the execution time of each map task, the number of map tasks of the corresponding job grows so quickly. This will cause each map slot to execute several map tasks, and the time for map tasks initialization and scheduling is much longer than the execution time of each map task. Thus, the query time is mainly decided by the number of map tasks that each map slot should execute.

7 Conclusions

In this paper, we present *TNBGR*, a system developed and optimized based on HBase and MapReduce to support multiple-field based queries for telecom applications in China. The experiment study shows significant performance advantage of our system. *TNBGR* can support major real world applications with telecom

CDR data, and our system is being deployed with a major telecommunication company in China.

Acknowledgments. This research was partially supported by the grants from the Natural Science Foundation of China (No.61070055, 91024032, 91124001);the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University(No. 11XNL010); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204).

References

1. Agrawal, P., Silberstein, A., Cooper, B., Srivastava, U., Ramakrishnan, R.: Asynchronous view maintenance for vlsd databases. In: SIGMOD 2009, pp. 179–192. ACM (2009)
2. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
3. Ding, L., Qiao, B., Wang, G., Chen, C.: An efficient quad-tree based index structure for cloud data management. In: Wang, H., Li, S., Oyama, S., Hu, X., Qian, T. (eds.) WAIM 2011. LNCS, vol. 6897, pp. 238–250. Springer, Heidelberg (2011)
4. Chang, F., Dean, J., Ghemawat, S., et al.: Bigtable: A distributed storage system for structured data. In: OSDI 2006, pp. 205–218 (2006)
5. Kellerman, J.: Hbase: Structured storage of sparse data for hadoop (2009), <http://hbase.apache.org/>
6. Kennedy, J.: Ithbase (2012), <https://github.com/hbase-trx/hbase-transactional-tableindexed>
7. Papadopoulos, A., Katsaros, D.: A-tree: Distributed indexing of multidimensional data for cloud computing environments. In: CloudCom 2011, pp. 407–414 (2011)
8. Wang, J., Wu, S., Gao, H., Li, J., Ooi, B.: Indexing multi-dimensional data in a cloud system. In: SIGMOD 2010, pp. 591–602. ACM (2010)
9. ykulbak. Ithbase (2012), <https://github.com/ykulbak/ithbase>
10. Zhang, X., Ai, J., Wang, Z., Lu, J., Meng, X.: An efficient multi-dimensional index for cloud data management. In: CloudDB 2009, pp. 17–24. ACM (2009)
11. Zou, Y., Liu, J., Wang, S., Zha, L., Xu, Z.: CCIndex: A complemental clustering index on distributed ordered tables for multi-dimensional range queries. In: Ding, C., Shao, Z., Zheng, R. (eds.) NPC 2010. LNCS, vol. 6289, pp. 247–261. Springer, Heidelberg (2010)

ST-HBase: A Scalable Data Management System for Massive Geo-tagged Objects

Youzhong Ma, Yu Zhang, and Xiaofeng Meng

School of Information, Renmin University of China, Beijing, China
`{mayouzhong,zhangyu1990,xfmeng}@ruc.edu.cn`

Abstract. In this paper, we propose ST-HBase (spatio-textual HBase) that can deal with large scale geo-tagged objects. ST-HBase can support high insert throughput while providing efficient spatial keyword queries. To the best of our knowledge, the existing approaches that deal with spatial keyword queries mainly focus on the static and medium-sized objects collections and cannot provide high insert throughput. In ST-HBase, we leverage an index module layered over a key-value store. The underlying key-value store enables the system to sustain high insert throughput and deal with large scale data, the index layer can provide efficient spatial keyword query processing. We propose two kinds of index approaches in ST-HBase: Spatial and Textual Based Hybrid Index(STbHI) and Term Cluster Based Inverted Spatial Index(TCbISI) which are suitable for different scenarios. We implement a prototype based on HBase that is a standard open-source key-value store. Finally we perform comprehensive experiments and the results show that ST-HBase has good scalability and outperforms the state-of-the-art approaches in terms of update and query performance.

Keywords: Spatial keyword query, Geo-tagged objects, HBase.

1 Introduction

With the development of positioning technology and the widely usage of smartphones, many objects on the web are being geo-tagged, such as the images in the Flickr, the tweets from Twitter, many spatial objects are also being associated with text descriptions. The combination of location and text results in a new kind of query: spatial keyword query. A spatial keyword query is like the follows: given a location or location area and a set of keywords, return a set of objects that satisfy the spatial requirements and are relevant to the query keywords. Such as finding the restaurants selling Peking Duck that are within 2 km from my current location or photos near by a given place whose text descriptions are similar to the query keywords.

Spatial keyword query processing has been studied in many literatures, the state-of-the-art approaches mainly employ a hybrid index combining R-tree index with textual inverted index. To the best of our knowledge, the existing R-tree based hybrid index mainly focuses on the static and medium-sized object collection. Although the hybrid index can improve the query performance by utilizing

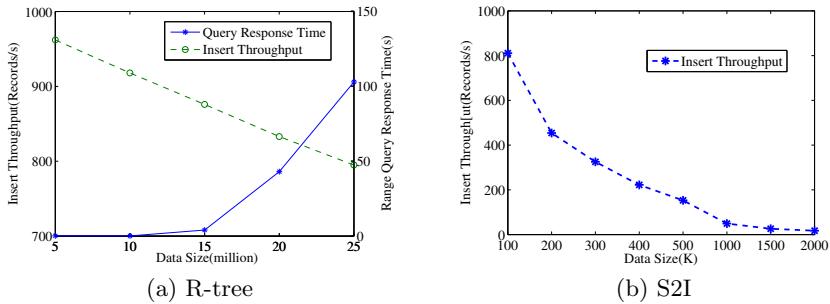


Fig. 1. Insert Throughput and Query Performance vs. Data Size

the spatial and textual pruning strategy, when the number of the targeted geo-tagged objects becomes very large, the overlap between R-tree nodes increases which make the query performance decrease dramatically.

Figure 1 shows the insert throughput and query performance of the R-tree like index at different data size. Figure 1-(a) shows the performance of R-tree index, it just uses R-tree to index the spatial information but neglects the textual information. Figure 1-(b) shows that the performance of the Hybrid index S2I[1]. S2I[1] took both spatial and textual information into consideration and proposed a hybrid index. From Figure 1, we can observe that the insert throughput is very slow and decreases with the data size increasing, because the R-tree like index needs additional cost to adjust the R-tree node and the inverted index file; the response time increases with data size increasing, because the overlap between the R-tree nodes increases as the data size becomes bigger and bigger.

In this paper, we propose spatial textual HBase(ST-HBase) which has good scalability and is capable of dealing with large scale dataset, ST-HBase can also support high insert rates and efficient spatial keyword queries. In ST-HBase, we use HBase to store the geo-tagged objects and combine spatial textual index with HBase, we propose two different approaches: Spatial and Textual Based Hybrid Index(STbHI) and Term Cluster Based Inverted Spatial Index(TCbISI). In summary, the main contributions of the paper are:

- We propose spatial textual HBase(ST-HBase) which is a scalable data management system. It has good scalability and can deal with huge amounts of geo-tagged objects;
- We propose two spatial textual index approaches: Spatial and Textual Based Hybrid Index(STbHI) and term cluster based inverted spatial index(TCbISI). We implement these two index approaches based on HBase;
- We implement a prototype using HBase and perform comprehensive experiments to exploit the scalability and efficiency of ST-HBase.

The rest of the paper is organized as follows. In section 2, we review the related works about spatial keyword queries; in section 3, we give the problem statement;

section 4 and 5 describe Spatial and Textual Based Hybrid Index(STbHI) and term cluster based inverted spatial index(TCbISI) respectively; in section 6, we perform detailed experiments and section 7 concludes this paper.

2 Related Work

Many research works have been done to exploit spatial keyword queries problem. Zhou et al. [2] were the pioneers to try to improve the performance of spatial keyword queries in search engines. They proposed three approaches by combing the inverted index and R-trees:(1) Building both R-trees and inverted indexes for the web documents from spatial and textual perspectives respectively; (2)Building the inverted indexes firstly and then creating a R-tree for every distinct term; (3)Creating a R-tree index for all the web documents and then creating one inverted index for the documents that are contained in each R-tree leaf node. Their experiments showed that the second approach achieved better performance.

Cong et al. [3] proposed a new hybrid indexing framework for top-k spatial keyword queries, the index framework combines the inverted indexes for text retrieval and the R-tree for spatial query. Several index approaches were explored within the framework. In the baseline approach, they index the web documents with R-tree and simply attach the inverted index to each R-tree node to obtain an Inverted file R-tree called IR-tree. For each leaf node, an inverted index is created for the documents that are contained in the leaf node, while the inverted index for each internal node represents all the documents in the sub-trees of the internal node. In addition to the baseline method, Cong et al. also proposed other two advanced approaches: DIR-tree and CDIR-tree. In DIR-tree, they incorporated document similarity when computing Minimum Bounding Rectangles(MBR) and made sure that the textual descriptions of the objects that are in the same R-tree node are also similar. In CDIR-tree, Cong et al. clustered the documents attached to spatial objects and employed a pseudo-document to represent each cluster. So more tighter and precise textual relevance bound can be estimated and the query performance can be further improved.

Rocha-Junior et al. [1] proposed another novel method that can process top-k spatial keyword query more efficiently. Differently from [3], Rocha-Junior et al. proposed a new index structure called Spatial Inverted Index (S2I). They adopted different organization methods for the terms based on their frequency. It is well known that the distribution of terms is very skewed, the document frequency of terms in a corpus follows the Zipf's law, which means that only a small number of terms occur frequently, while most of the terms occur infrequently[1][4]. So S2I mapped each more frequent term to a distinct aggregated R-tree (aR-tree) that stores the objects with the given term, if the number of the objects corresponding to one given term does not exceed a given threshold, the objects are just stored in a file. Based on the S2I index, they also proposed two efficient algorithms(Single Keyword Algorithm: SKA and Multiple Keyword Algorithm: MKA) to process the top-k spatial keyword queries efficiently.

There are many other approaches that focus on spatial keyword queries. Ian De Felipe et al. [5] combined R-tree and signature file, and proposed a new

efficient hybrid index called IR²-tree(information retrieval R-tree). Each node of IR²-tree contained two parts of information: space information and keywords information. Zhang et al. [6] proposed bR*-tree to process m -closet keyword queries; Zhang et al. [7] proposed a Light-Weighted Index to process the spatial keyword queries for the objects that are described by tags.

3 Problem Statement

In this section, we briefly describe the problem addressed in the paper. Let D be a geo-tagged database. Each geo-tagged object O in D can be represented as a triple $(O.id, O.loc, O.doc)$, in which $O.id$ is the identifier of the object, $O.loc$ represents the location of the object and contains latitude and longitude, $O.doc$ is the text that describes the object. Table 1 is an example of D .

Table 1. Geo-tagged Objects Database

$O.id$	$O.loc$	$O.doc$
o_1	(111.12, 108.23)	{apple banana }
o_2	(112.21, 108.12)	{noodle pizza }
o_3	(111.12, 110.21)	{Peking duck }
o_4	(113.21, 110.54)	{great wall }
o_5	(112.32, 109.65)	{coca cola }

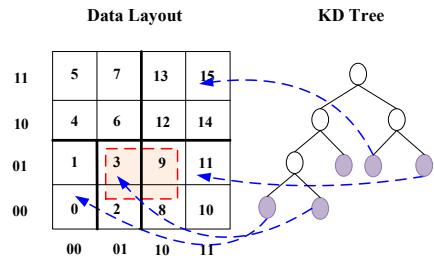


Fig. 2. Spatial Index

Given a spatial keyword query $Q(Q.loc, Q.keywords, \sigma)$, in which $Q.loc$ is a given location, $Q.keywords$ is the query keywords and σ is the distance threshold, we need to find out all the geo-tagged objects which are far away from the location $Q.loc$ within σ and contain the query keywords. After we obtain all the objects that satisfy the textual and spatial constraints of the query, we compute the relevance scores for all the objects and rank the objects according to their scores. The top-k spatial keyword query can be finished by extending the space range of the query incrementally. In this paper we mainly focus on the range query processing.

4 Spatial and Textual Based Hybrid Index

4.1 Overview of STbHI

In this paper we want to manage the large scale geo-tagged objects using HBase. It is well known that HBase organizes the data based on the rowkeys and can provide very efficient queries on rowkeys. We try to design a suitable rowkey generation scheme that can combine the textual and spatial information of the geo-tagged objects together, so that we can deal with the spatial keyword queries more efficiently.

Inverted Index Table Schema. In order to implement textual filtering, we build the inverted index. While different to the traditional inverted index file, we use a HBase table called "Inverted Index Table" to finish the same task. Table 2 displays the scheme of the inverted index table. The rowkey is the composition of the term and the z-ordering value of the object. t_i represents the i_{th} term, id_m is the object id, z_m (e.g., 01011010) is the z-ordering value of id_m , (x_m, y_m) represents the latitude and longitude of the object respectively. The z-ordering value of each geo-tagged object can be computed based on its location information(latitude and longitude), because z-ordering technique preserves the original locality of the objects and the z-ordering value of the objects that are nearby in the original space are likely close to each other. The advantage of such kind of design is that we can implement the textual filtering and spatial filtering simultaneously and translate a spatial keyword query into several range queries on the rowkey

KD-Tree Index. As long as we encode the objects using z-ordering technique, we can translate the spatial query into range query on the z-ordering values. For instance, given a query rectangle $R[(x_{min}, y_{min}), (x_{max}, y_{max})]$, we can get the minimum z-ordering value z_{min} and the maximum z-ordering value z_{max} and then execute a range query by using $[z_{min}, z_{max}]$. But in some cases, it is not feasible if we directly execute the range query from the query rectangle, as it will produce many false positives. Just as shown in Figure 2, the red bordered rectangle is the range query, and the range of the z-ordering value is [2, 9]. If we directly execute the range query [2,9], we have to scan every object from 2 to 9. But actually what we want to get just include 2, 3, 8 and 9, the objects from 4 to 7 are false positives. In order to eliminate the false positives as many as possible, we partition the objects by using KD-tree index, just like Figure 2 depicts, for the same query rectangle, we obtain two subspaces and get the required objects 2, 3, 8 and 9 finally.

Table 2. Inverted Index Table Schema

rowkey	col. fam.: cf		
	col: id	col: x	col: y
$t_101011010$	id_1	x_1	y_1
$t_101011011$	id_2	x_2	y_2
$t_211011010$	id_3	x_3	y_3
$t_301011010$	id_1	x_1	y_1
$t_411011010$	id_3	x_3	y_3

21	23	29	31	53	55	61	63
20	22	28	30	52	54	60	62
17	19	25	27	49	51	57	59
16	18	24	26	48	50	56	58
5	7	13	15	37	39	45	47
4	6	12	14	36	38	44	46
1	3	9	11	33	35	41	43
0	2	8	10	32	34	40	42

Fig. 3. Query Processing

The procedure of the data insertion is as follows. Given a geo-tagged object $O.id, \langle O.lat, O.lon \rangle, \{t_1, t_2, t_3, t_2\}$, firstly we compute the z-ordering value of $O.id$, represented as $Z_{O.id}$. Then insert the object $O.id$ into KD-tree, and decide whether the corresponding subspace needs to split or not, if the data size of a subspace exceeds the give threshold, the subspace needs to split; (This step is

optional, if the KD-tree is built beforehand, this step can be omitted.) Finally we construct the inverted index data rows and insert them into Inverted Index Table: $(t_1 Z_{O.id}, 1, O.id)$, $(t_2 Z_{O.id}, 2, O.id)$, $(t_3 Z_{O.id}, 1, O.id)$.

4.2 Query Processing

Given a spatial keyword query $Q(Q.loc, Q.keywords, \sigma)$, in which $Q.loc$ is a given location, $Q.keywords$ is the query keywords, σ is the distance threshold, $Q.loc$ and σ can be represented as a rectangle area $R[(Q.loc_x - \sigma, Q.loc_y - \sigma), (Q.loc_x + \sigma, Q.loc_y + \sigma)]$. When dealing with the spatial keyword query, we need to choose the suitable query strategy according to the area of the query rectangle, it can be divided into the following three categories:

- As shown in Figure 3, if the area of the query rectangle is relative small, just like the red bordered rectangle, it just covers four distinct z-ordering values(12, 13, 14, 15), we can directly compute the z-ordering values of all the objects contained in the rectangle, then access the Inverted Index Table according to query keywords and the z-ordering values.
- As the area of the query rectangle becomes larger, the number of the distinct z-ordering value also increases, in such case, if we still compute each z-ordering value and query them from the Inverted Index Table one by one, it will cost too much time. So we can figure out the minimum and maximum z-ordering value of the query rectangle, then obtain the z-ordering value range, if the range is smaller than a given threshold, we can use the keywords and the z-ordering value range to request from the Inverted Index Table. In such case, there will be some false positives, just like the blue bordered rectangle, the minimum and maximum of the z-ordering value the rectangle covers are 2 and 14 respectively, what we actually want to get are 2, 3, 6, 8, 9, 10, 11, 12, 14 and 4, 5, 7, 13 are the false positives. But the number of the false positive is small, we can still gain more benefits.
- While if the z-ordering range of the query rectangle exceeds a given threshold, there will be much more false positives, in such cases, we need to resort to the KD-tree index. We can obtain the related subspaces by querying the KD-tree index using the query rectangle, then compute the desired ranges according to the area of the regions and the query rectangle. Just like the green border rectangle, by querying KD-tree index we can get three subspaces whose z-order value ranges are 12-15, 16-31 and 32-63 respectively. Supposing that the range of the query rectangle is 14-59, we get the intersects of the query range and each subspace, and the desired z-ordering value ranges are 14-15, 16-31 and 32-59, many false positives have been eliminated.

4.3 Pre-Splitting for Inverted Index Table

It is well known that HBase organizes the data as regions and the data is stored in regions according to their rowkey. At the beginning, all the data is inserted into just one region, if the number of the records in the first region exceeds the given

threshold, the region will split into two roughly equal sized child regions. As more and more data is inserted, the regions will split recursively. Such scheme has two pitfalls, one is the concurrency, the other one is the high splitting cost. Because all the data are inserted into one region or several regions at the beginning, the concurrency will be constrained by the limited regions, in order to enhance the concurrency, we can pre-split the regions for the Inverted Index Table.

In Inverted Index Table, the rowkey is composed of term and the z-ordering value of the geo-tagged objects, so the records will be stored into different regions according to the terms and the location information of the objects. We all know that, the distribution of the term frequency is always skewed and obeys to the *Zipf*-like distribution, what we want to do is that we can divide all the distinct terms into several segments, and make sure that the sum of the term frequency in each segment is roughly equal. The detail of the pre-splitting is described in Algorithm 1.

Algorithm 1. Pre-splitting for Inverted Index Table

Input: A geo-tagged object dataset $D = \{o_1, o_2, \dots, o_N\}$;
 the number of the pre-splitting regions K ;
 the length of the z-ordering value L ;
Output: K splitting keys: $S = \{\hat{t}_1 0_1 0_2 0_3 \dots 0_L, \hat{t}_2 0_1 0_2 0_3 \dots 0_L, \dots, \hat{t}_K 0_1 0_2 0_3 \dots 0_L\}$

- 1: Building a subset $\hat{D} = \{o_1, o_2, \dots, o_M\}$ by sampling from the original dataset D
 - 2: Calculate the frequency of each unique term and sort the terms in descending order:
 $T = \{(t_1, f_1), (t_2, f_2), \dots, (t_V, f_V)\}$
 - 3: Calculate the sum of the term frequency for all the terms: S
 - 4: $S \leftarrow \emptyset$
 - 5: $temp \leftarrow 0$ //temporal variable used to compute the sum of the frequency
 - 6: **while** ($T \neq NULL$) **do**
 - 7: get the first term t_i and its frequency f_i from T
 - 8: $temp \leftarrow temp + f_i$
 - 9: remove (t_i, f_i) from T
 - 10: **if** $((\lfloor \frac{S}{K} \rfloor - temp) \leq \sigma)$ **then**
 - 11: add $\hat{t}_i 0_1 0_2 0_3 \dots 0_L$ into S
 - 12: $temp \leftarrow 0$
 - 13: **end if**
 - 14: **end while**
 - 15: return the pre-splitting key set S
-

5 Term Cluster Based Inverted Spatial Index

In STbHI index approach, for a given geo-tagged object o_i , if o_i has n terms, then we have to insert one index entry for each term into the Inverted Index Table. That is to say, we have to do n times insertion for each geo-tagged object o_i , so the index speed will decrease and the space overhead will be high. At the same time, when we deal with a spatial keyword query Q that contains m query keywords, we have to execute m sub-queries and then combine the results of all the sub-queries, so the query performance will also be affected. Because of

the above reasons, STbHI Index approach is not the best suitable solution for managing the geo-tagged objects that contain many terms and dealing with the multi-keywords queries. In order to tackle the above problem, we propose another index solution: Term Cluster Based Inverted Spatial Index(TCbISI). This idea is mainly inspired by the following observations: some terms are usually used together to describe the same objects and some terms often occur in the same query. According to the above observation, we can divide the terms into different clusters based on the co-occurrence relationship among the terms, the objects that correspond to one term cluster can be treated together, each cluster is mapped to a HBase table. The objects of a term cluster are stored in one HBase table and the organization of the objects is like that of STbHI. By doing so, when we deal with a query that contains several keywords, the number of the HBase tables that we need to access is less than the number of the keywords, so the response time can be decreased.

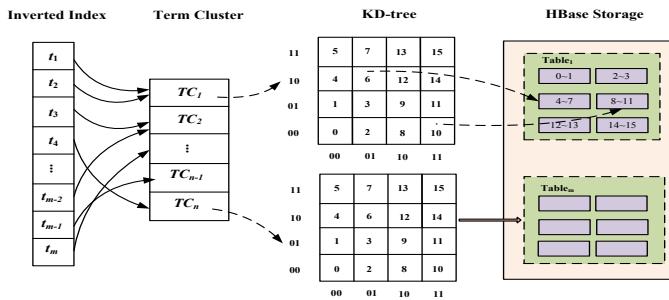


Fig. 4. Framework of Term Cluster based Inverted Spatial Index

5.1 Framework of Term Cluster Based Inverted Spatial Index

Figure 4 shows the framework of Term Cluster based Inverted Spatial Index. The TCbISI index consists of four components: Inverted Index, Term Cluster Index, KD-tree and HBase Table.

- **Inverted Index.** The Inverted Index is used to implement textual pruning in which each term is mapped to a term cluster. Although the geo-tagged objects are unlimited, the total number of the distinct terms is always limited, we can suppose that the inverted index can be stored in the main memory.
- **Term Cluster Index.** Term Cluster Index is mainly used to map the term cluster to the KD-tree or HBase table, the number of the term clusters is just several thousand, so the term cluster index can also be stored in main memory.
- **KD-tree.** The KD-tree is used to divide the geo-tagged objects that corresponds to each term cluster into several partitions based on the z-ordering value of the objects, when dealing with spatial keyword queries, we can quickly located the related HBase regions using KD-tree.

- **HBase Table.** HBase Table is the actual storage of the objects, it uses the z-ordering value of the object as its rowkey. So the objects that are close in the original space are likely to be stored in the same HBase region.

5.2 Query Processing

According to the framework of TC_bISI we can finish a spatial keyword query in several steps: given a spatial keyword query $Q(Q.loc, Q.keywords, \sigma)$, firstly, getting the term clusters which contain $Q.keywords$ through Inverted Index; Secondly obtaining the corresponding KD-trees or HBase table names through Term Cluster Index; Thirdly adopting suitable query methods according to the query range; Finally we merge the results returned from each term cluster and obtain the final results. The remaining query procedure is the same as that of ST_bISI.

5.3 Term Cluster Generation

Term cluster is the basis of TC_bISI, in order to obtain the high quality clustering result, we propose a Term Co-occurrence Graph based clustering method(TCG-Cluster), in which each node represents a term and the edge between two nodes represents the co-occurrence frequency of two terms. Because of the space limit, the detail of TCG-Cluster will not be listed here and it can be referred from the extended version of the paper.

6 Experimental Evaluations

In this section, we compare our approaches with the *Spatial Inverted Index*(S2I) proposed by Joao et al.[1], to the best of our knowledge, S2I is the state-of-the-art approach that addresses spatial keyword queries. The setup for S2I is the same as described in [1], the nodes of the aR-trees have a block size of 4KB and are able to store between 42 and 85 entries.

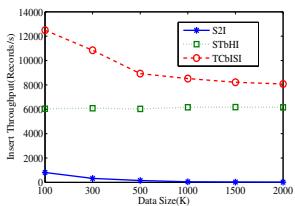
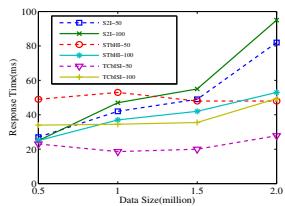
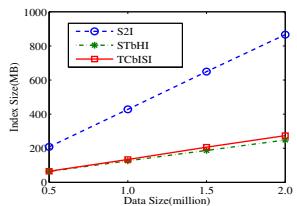
6.1 Experimental Settings

Setup. The experiments for S2I were executed on a server with a 3GHz Dual Core AMD processor and 4GB memory. Our approaches were implemented on HBase-0.20.6 and Hadoop-0.90.4, the cluster size is 16 nodes that are connected with 1Gbit Ethernet switch. The main parameters and values used in the experiments are described in Table 3. The default values are presented in bold.

Datasets. Table 3(b) shows the characteristics of the datasets used in the experiments. In this paper we mainly focus on querying from large scale geo-tagged objects, we have two real datasets presently, one is twitter dataset containing about 10,000,000 tweets, another one is a document corpus from Reuters containing 10,000 documents. Based on the two real datasets we generated several other datasets, the detailed statistics of them are listed in Table 3(b).

Table 3. Experimental Settings

(a) Experimental Setup		(b) Characteristics of the datasets			
Parameter	Values	Datasets	Tot. no. of objects	Avg. no. of unique terms/obj	Tot. no. of unique terms
Num. of keywords	1, 2, 3 , 4, 5	Twitter5	0.5M	6.42	226,969
Spatial range	5, 50, 100 ,	Twitter10	1M	6.44	380,727
Datasets(M)	200, 300, 500	Twitter15	1.5M	6.47	515,676
Comuter Node	0.5,1,1.5,2,100	Twitter20	2M	6.46	648,518
	4, 8, 12, 16	Data-Syn	100M	9.07	1,485,225

**Fig. 5.** Insert Throughput vs. Data Size**Fig. 6.** Query Performance vs. Data Size**Fig. 7.** Index Space Requirement vs. Data Size

6.2 Performance Varying the Data Size

Insert throughput. Figure 5 shows that the insert throughput of S2I, STbHI and TCbISI with different data size. We can see that the insert throughput of S2I is the worst one, and the insert throughput decreases dramatically with the data size increasing. That is because S2I adopts the R-tree like index and R-tree like index needs additional cost to keep the tree balanced which affects the insert performance. The insert performance of TCbISI is better than that of STbHI, the main reason is as the follows: a given geo-tagged object o_i containing m keywords needs m times insertion for STbHI approach, while the insertion times of TCbISI is equal to the number of the clusters which the keywords of o_i belong to, so the insertion times of TCbISI is less than that of STbHI. We can also observe that the insert throughput of STbHI and TCbISI has good scalability, they can keep high insert throughout with the data size increasing. The peak insert throughput of TCbISI is over 10,000 objects per second.

Query Performance. Figure 6 mainly describes the query performance of the different index approaches. When the data size and the query range are both relative small(e.g., the data size is less than about 1,500,000 and the query range is smaller than 50), S2I can obtain the better query performance than STbHI. While as the data size increases, the query performance of S2I is becoming

more inferior than that of STbHI-50 and TCbISI-50. For the large range query, the query performance of STbHI and TCbISI is always better than that of S2I, e.g., STbHI-100 and TCbISI-100.

Index Space Requirement. Figure 7 shows the storage space required for S2I, STbHI and TCbISI. The size required by S2I is larger than the size required by STbHI and TCbISI. The reason is that STbHI and TCbISI are both implemented on HBase, while the data in HBase is actually stored on HDFS and HDFS always adopts many compression algorithms to improve the space utilization. In order to keep fault tolerant, HDFS always keep 3 replicas for each data block, even so, the total size requires is still smaller than that of S2I.

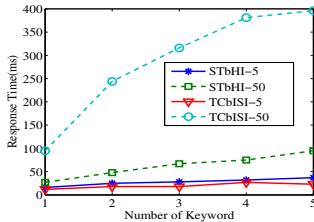


Fig. 8. Response Time vs. Num. of Keyword words

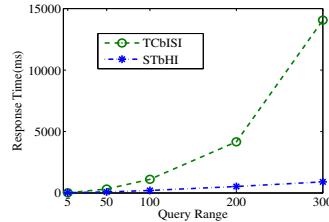


Fig. 9. Response Time vs. Query Range

6.3 Varying the Number of Keywords

Figure 8 depicts the response time while varying the number of keywords (from one keyword to five keywords) for large scale dataset (100M geo-tagged objects), the default query range is set to 0-50. From section 6.2, we know that the time consumed by S2I for inserting two million objects is about 32 hours, it will spend more than 66 days to insert 100,000,000 objects, so in this section we just test the response time of STbHI and TCbISI. The response time of STbHI increases almost linearly with the number of the keywords. For the query with small range, the query performance of TCbISI is better than that of STbHI, while for the large range query, the performance of STbHI is much better than that of TCbISI. The main reason is as the follows: for the large range query, many records need to be scanned and verified, in STbHI we can finish the filtering and verification based on the rowkey, so that is very fast. While in TCbISI, we need additional time to filter and verify the records on the non-rowkey column.

6.4 Varying the Query Range

In this section we mainly test the query response time of STbHI and TCbISI for the queries with different range varying from 5, 50, 100 to 200, 300, 500, the query contains three keywords, the number of geo-tagged objects is the same as in section 6.3. In order to obtain the relative accurate response time, We execute ten times queries and use the average time as the final response time.

As the query range increases, more objects need to be scanned and transformed to the client side, so the response time increases as the query range becomes larger. For the small range queries(≤ 5), the performance of ST_bHI is better than that of TC_bISI, as the query range increases, the performance of ST_bHI is becoming much better than that of TC_bISI, the reason is as the same described in the above subsection.

7 Conclusions and Future Work

In this paper, we propose a scalable data management system for massive geo-tagged objects called ST-HBase (Spatial Textual HBase) in which we use HBase as the storage so that it can provide high insert throughput. In order to support efficient spatial keyword query, we layer KD-tree index on HBase and propose two kinds of index approaches. In the future, we plan to extend our work to ad-hoc spatial temporal analysis about the large scale geo-tagged objects, because many geo-tagged objects also contain time dimension, such as photos in Flickr, tweets from Twitter, and the time dimension is also important for analyzing the geo-tagged objects, for instance we can compute the temporal and spatial distribution of the tweets about some specific topics.

Acknowledgments. This research was partially supported by the grants from the Natural Science Foundation of China (No. 61070055, 91024032, 91124001); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University(No. 11XNL010); National 863 High-tech Program (2012AA010701, 2013AA013204).

References

1. Rocha-Junior, J.B., Gkorgkas, O., Jonassen, S., Nørvåg, K.: Efficient processing of top-k spatial keyword queries. In: Pfoser, D., Tao, Y., Mouratidis, K., Nascimento, M.A., Mokbel, M., Shekhar, S., Huang, Y. (eds.) SSTD 2011. LNCS, vol. 6849, pp. 205–222. Springer, Heidelberg (2011)
2. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.Y.: Hybrid index structures for location-based web search. In: CIKM 2005, pp. 155–162 (2005)
3. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top-k most relevant spatial web objects. In: PVLDB, pp. 337–348 (2009)
4. Joachims, T.: A statistical learning model of text classification for support vector machines. In: SIGIR 2001, pp. 128–136 (2001)
5. Felipe, I.D., Hristidis, V., Rishe, N.: Keyword search on spatial databases. In: ICDE 2008, pp. 656–665 (2008)
6. Zhang, D., Chee, Y.M., Mondal, A., Tung, A.K.H., Kitsuregawa, M.: Keyword search in spatial databases: Towards searching by document. In: ICDE 2009, pp. 688–699 (2009)
7. Zhang, D., Ooi, B.C., Tung, A.K.H.: Locating mapped resources in web 2.0. In: ICDE 2010, pp. 521–532 (2010)

EasyControl: Improve Database Throughput under Overloading^{*}

Chengkai Tao, Qizhi Liu, and Yan Zhang

State Key Laboratory for Novel Software Technology, Nanjing University, China
tck@live.cn, {lqz, zycs}@nju.edu.cn

Abstract. When facing big data, overloading in database usually occurs, leading to the decline in performance. This is a problem that has been studying for long but still not completely solved yet. Preventing overloading while fully utilize resources is challenging. In this paper, we present EasyControl, a prototype admission control system that reduces the impact of overloading. The most significant distinction between EasyControl and existing overloading controls is that the users don't need to study the system beforehand. Experiments show that adding EasyControl increases the throughput during overloading periods and will not affect the throughput when the load is mild.

1 Introduction

Big data usually includes some large data sets and complex queries that exceed the capacity of the system. When dealing with this kind of workload, the performance usually decreases and the system becomes overloaded. If not handled properly, it may lead to a disastrous deterioration of both transaction throughput and response time.

To prevent the database from overloading, one of the most important configuration parameter is the multi-programming level (MPL). The MPL determines the maximum number of transactions can be concurrently processed in the database. Setting the MPL is no easy task [1]: if it is set too high, the system becomes susceptible to thrashing; if it is set too low, not all resources are utilized and the performance will drop. What's more, the optimal MPL strongly depends on the workload characteristics which may change with time.

Many researches [2-8] have been taken to pursue good algorithms and tools for (semi-) automatic overloading control. Workload related information is exploited to find MPL in most approaches, except for [2] and [3]. However, the required workload characteristic is not always obtainable. For example, online retailers may have big sales on holidays. It is at this time the database systems are likely to be overloaded. But the workload is different from past normal days which makes it difficult to estimate. [2] and [3] give some golden standards in transaction processing systems: keep

^{*} This research was supported by the National Natural Science Foundation of China (No. 61170068), National Social Science Foundation of China (No. 11AZD121) and Opening fund of Shanghai Key Laboratory of Trustworthy Computing.

the conflict ratio below the critical threshold of 1.3, or fraction of blocked and mature transactions below 0.5. But whether it is the best value for the entire modern database systems remains unknown.

The problem we face is similar to the problem in industrial control systems. There is a delay (query processing time) between input settings (query request) and new settings take effect (query response). And the environment (algorithms in database systems, computing power, memory capabilities, etc.) is too complex to model. In the absence of knowledge of the underlying process, a PID controller is the best controller. A PID controller needs a fixed target point to guide the whole controlling process. But different systems may have different capabilities, no golden target point suitable for all can be found.

In this paper, we present EasyControl – a prototype admission control system for determining a suitable MPL automatically. Unlike existing works, the users do not need any prior knowledge about the workload, database system or hardware configuration. We think this “Easy” characteristic is especially important in today’s cloud environment, where the database in the cloud is just providing the service, not knowing what the users’ applications are, and can’t estimate the workload beforehand.

2 Related Work

In database systems, there are generally two categories of controlling methods: external (admission) control and internal control.

External control [4], [9] treats the database systems as black boxes. It determines the concurrency level by measuring the indicators outside the database systems. External control methods are usually adaptive, but maybe not as accurate as internal control. Internal control [2], [3] dives into the database systems and tries to find indicators inside. A typical indicator is the number of blocked transactions. Main advantage of internal control is more accurate on a per-component basis. But it usually considers a single factor (usually locking) alone, not the whole system.

EasyControl uses external control. According to our experiments, overloading of database is jointly caused by many factors. Controlling directly using the external throughput can take the combined effect into consideration. What’s more, setting an external controller over existing databases is much easier than altering the inside.

Control-theory and machine learning are also used in some previous works [7], [8] targeting at web servers. The process variable is usually the response time. The main innovation of our EasyControl algorithm is that it gets rids of an explicit control target value. Also, since the overloading situation doesn’t happen very often, there is little chance to train the system before it really happens.

3 EasyControl Algorithm

In this section we present our admission control algorithm used in EasyControl. The algorithm is based on the idea of PID controller, while it is improved and made suitable for the complex database overloading control environment.

3.1 Classical PID Controller

Proportional–integral–derivative controller (PID controller) is a well-known feedback controller. Its general idea is to minimize the error between the process value and the set point by setting system input based on present, past and predicted future errors. The controller output is calculated as the sum of three terms:

$$u = K_p e + K_i \int edt + K_d \frac{de}{dt} \quad (1)$$

The proportional term considers how far away the process value is from the set point and makes adjustment immediately. The integral term accumulates previous offsets to reduce the final error. The derivative term slows the rate of change and helps reduce overshoot and ringing. K_p , K_i and K_d are three tuning parameters representing proportional, integral and derivative gain respectively.

Although powerful, using PID controller as a database admission controller has some difficulties: (1) Different database systems and different workloads often show different characteristics, finding a set point suitable for all systems is also tricky; (2) PID tuning is usually done through some manual trial-and-error approaches and computer simulations. But in OLTP applications, data in the database is constantly changing. So it is hard to find three optimum parameter values.

To overcome these deficiencies of classical PID controller, we changed the way it is used and try to find a dynamic set point, which can represent the system's real-time capability. Next we outline our overloading prevention algorithm.

3.2 Admission Control Algorithm

The only two variables we measure directly are the number of user requests N_q and database responses N_s . And from these we derive another variable: the number of newly added pending requests N_p . It is calculated as the difference between request number and response number $N_p = N_q - N_s$.

Due to the variance in server capability and query complexity, there is no easy way to use direct indicators like throughput or response time as set point. So our set point is a balance between requests and responses: server response number equals user request number. Three terms in the PID controller are: response number in current measure interval, i.e. $K_p N_s$; total number of pending requests in database, i.e. $K_i \sum N_p$ and newly added pending request number in current measure interval, i.e. $K_d N_p$ respectively.

Tuning parameters is another difficult problem. But an advantage of PID controller is that it can provide acceptable control using default tunings. Therefore we simply choose naive parameters based on their physical meanings in database:

$u = N_s - \sum N_p - N_p$. Because of the naive parameters used, the output maybe inaccurate. We maintain a separate variable N_a - maximum allowed request number to do the admission control. If u is greater than 0, it suggests that the system is free to overload and can accept more requests, we increase N_a and vice versa.

3.3 EasyControl Implementation

Our admission controller is implemented as a proxy outside the database. Its existence is fully transparent to both user and database server. The architecture is depicted in Figure 1.

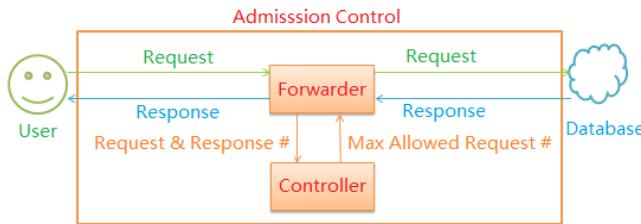


Fig. 1. EasyControl System Architecture

When the Forwarder receives a request from the user, it checks whether the request sent in current measure interval plus the number of pending requests exceeds the max allowed request number N_a . If so, the Forwarder delays the forwarding until next measure interval; otherwise it forwards the request and increase the request sent in this measure interval counter. Server's responses are always forwarded immediately.

The Controller executes the algorithm described in Section 4.2. After started it has an initialization period. During this time the max allowed request number N_a is initialized to the average request number measurements from the Forwarder. At this moment no actual control is done because the system is warming up and thus unstable. Once warmed up, in every measure interval a PID controller output u is calculated using the formula: $u = N_s - \sum N_p - N_p$. If $u > 0$ and the forwarded request number reaches the max allowed request number N_a , N_a is increased by multiplying (1+adjust factor); if $u < 0$, N_a is decreased by multiplying (1-adjust factor).

4 Experiments

Our experiments are conducted on two environments listed in Table 1. We use the TPC-C benchmark on MySQL 5.5. Test duration is set to 8 hours. Environment 1 is our main evaluation environment, 3 data scales is set from 100 data warehouses

(about 6.4 GB data) to 300 data warehouses (about 19 GB data). Under each data scale we test 10 different user numbers from 1 times data warehouse number to 10 times data warehouse number. In Environment 2, we compare our algorithm with two existing algorithms: Incremental Steps algorithm and Parabola Approximation algorithm proposed by Heiss and Wagner [4].

Table 1. Two Environment Setups in Experiment #2

No.	Configuration Combination
Environment 1	2.66GHz Core 2 Duo CPU, 3G DDR2 memory with a 320GB 7200 RPM SATA hard disk, Microsoft Windows Server 2008 R2 Data warehouse number used: 100, 200, 300 Concurrency level used: 100, 200, ..., 1000
Environment 2	3.3GHz Core i3 2120 CPU, 4G DDR3 memory with a 500G 7200 RPM SATA hard disk, Microsoft Windows 7 64 bits Data warehouse number used: 500 Concurrency level used: 1000

The total number of server responses given per second is used as our performance metric, as this is what optimized by the EasyControl algorithm, and should be proportional to the tpmC metric. There are three parameters in our EasyControl proxy: measure interval, initialization time and adjust factor. To make the tuning accurate and adaptive to most systems. We fix measure interval to 1 second, initialization time to 5 minutes and adjust factor to 0.01.

Figure 2 shows the comparison of throughput between with and without EasyControl in Environment 1. The throughputs are labelled as “EasyControl” and “No Control” respectively. We can see from the graph: (1) when the user load is low, MySQL with EasyControl gives almost identical throughput to naive MySQL; (2) when the user load is high, adding EasyControl significantly increases the throughput.

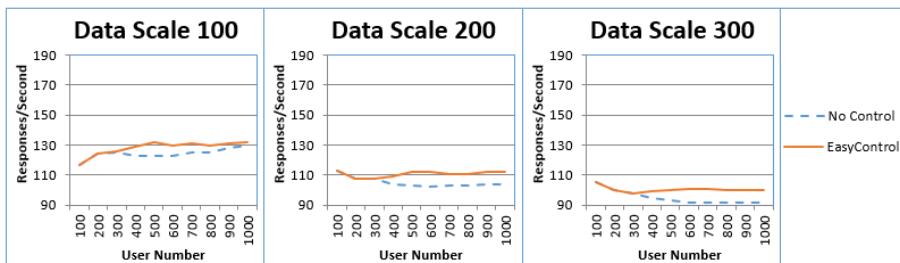


Fig. 2. Throughput Comparison of EasyControl and No Control

Incremental Steps algorithm and Parabola Approximation algorithm [4, 6] are implemented on MySQL and tested on Environment 3 together with EasyControl. One set of representative results are given in Table 4. EasyControl outperforms the other two algorithms. We think the fine-grained control interval of EasyControl contributes most to its advantage. As stated in [6] and verified by us, to avoid the influence of

potentially noisy measurements, we can only set the control interval of IS and PA to 60 seconds, while it is only 1 second in EasyControl. PA even fails to improve the original result because of the output MPL is too sensitive to normal, minor workload fluctuations. Maybe it needs an even larger control interval to avoid this problem.

Table 2. Comparision with IS and PA Algorithms (1000 users and 500 warehouses)

Control Method	No Control	EasyControl	IS	PA
Responses per Second	95.26	102.31	96.58	94.69

5 Discussion and Conclusion

In this paper, we studied the effects of overloading on the performance of relational databases. Even though relational database is considered not suitable for big data in many cases, it is still the most widely used database system. Some non-relational and high performance systems may have much greater capabilities, but the capabilities are not infinite. The systems still have the possibilities of being overloaded. We think the general approaches in dealing overloading can be extended to similar systems.

To reduce the impact of overloading, we presented EasyControl. EasyControl uses an idea similar to PID controller that considers pending requests and previous throughput to judge current databases' status. We experimentally demonstrated that without any tuning EasyControl can improve overloaded databases' performance. For future work, we are exploring the combination of different workload on EasyControl and its effect on real production databases.

References

1. Weikum, G., Moenkeberg, A., Hasse, C., Zabback, P.: Self-tuning database technology and information services: from wishful thinking to viable engineering. In: VLDB 2002 (2002)
2. Monkeberg, A., Weikum, G.: Performance evaluation of an adaptive and robust load control method for the avoidance of data-contention thrashing. In: VLDB 1992 (1992)
3. Carey, M.J., Krishnamurthi, S., Livny, M.: Load Control for Locking: the Half-and-Half Approach. In: PODS 1990 (1990)
4. Heiss, H.-U., Wagner, R.: Adaptive load control in transaction processing systems. In: VLDB 1991 (1991)
5. Thomasian, A.: Thrashing in Two-Phase Locking Revisited. In: ICDE 1992 (1992)
6. Abouzour, M., Salem, K., Bumbulis, P.: Automatic tuning of the multiprogramming level in Sybase SQL Anywhere. In: International Workshop on Self-Managing Database Systems 2010 (2010)
7. Abdelzaher, T., Shin, K.G., Bhatti, N.: Performance guarantees for Web server end-systems: A control-theoretical approach. IEEE Transactions on Parallel and Distributed Systems 2002 (2002)
8. Tozer, S., Brecht, T., Aboulnaga, A.: Q-cop: Avoiding bad query mixes to minimize client timeouts under heavy loads. In: ICDE 2010 (2010)
9. Schroeder, B., Harchol-Balter, M., Iyengar, A., Nahum, E., Wierman, A.: How to determine a good multi-programming level for external scheduling. In: ICDE 2006 (2006)

Combination of In-Memory Graph Computation with MapReduce: A Subgraph-Centric Method of PageRank

QiuHong Li¹, Wei Wang¹, Peng Wang¹, Ke Dai¹,
ZhiHui Wang^{1,*}, Yang Wang¹, and Weiwei Sun²

¹ Fudan University

² Institute of Computer Science & Technology, Peking University
{09110240012, weiwang1, pengwang5, daike, zhhwang, wy}@fudan.edu.cn,
weiweisun@pku.edu.cn

Abstract. In order to improve the efficiency of the PageRank algorithm, parallelizing methods, especially the ones based on MapReduce, interest many researchers during the past several years. Previous implementations of the PageRank algorithm on MapReduce ignore the characteristic of locality in distributed systems which is very important to reduce the I/O and network costs. In this paper, we explore the locality property and propose a new method for fast PageRank computation by supporting a subgraph as an input record for map functions. Graph partitioning techniques and a message grouping method are employed to guarantee the efficiency of communication among different subgraphs. Experiments show that our method is significantly more efficient than previous approaches without accuracy loss. The key idea to change the granularity of basic processing units from edges to subgraphs can benefit many other parallelizing algorithms for graph processing.

1 Introduction

In many applications such as web data management and social network, we always meet many massive graphs. But scalability is a problem for traditional in-memory graph algorithms. A promising platform for massive graph algorithm is Hadoop [1] which provides MAPREDUCE [2] computation model for distributed computing. In this paper we take PageRank [3] as an example to explore how to combine in-memory graph algorithm with MapReduce. To get a high scalability system many researchers focus on efficiently parallelizing PageRank on MapReduce. But there are many challenges to implement the pageRank algorithm on MapReduce, such as data partition, reduce the communication cost. Moreover, as a typical graph processing algorithm, techniques developed to improve the efficiency of the PageRank algorithm can be easily extended to resolve many other graph-based algorithms.

Previous implementations of the PageRank algorithm on MapReduce ignore the characteristic of locality in distributed systems which is very important to reduce the I/O and network costs. Broadly speaking, in a distributed network without shared memory, processes cooperate by exchanging messages. Since sending messages to far away

* Corresponding author.

nodes is expensive, computation should be based on local information as much as possible. Both HaLoop [4] and PEGASUS [5] provide implementations of PageRank on MapReduce. But the computation process are implemented by two rounds of MapReduce jobs for one PageRank iteration which cause extra I/O costs. Previous approaches are edge-centric and produce a large amount of internal results which lead to high Shuffle Bytes for Hadoop job. In this paper, we explore the locality property and propose a new subgraph-centric method for fast PageRank computation only using one MapReduce job for one PageRank iteration by the adoption of *map* cache and graph partitioning strategies. We especially consider decreasing the communication cost in the cluster and improving the local throughput of the PageRank algorithm for distributed systems. The contributions of this paper are as follows:

1. We propose a method to adapt MapReduce for graph applications. We treat a subgraph as an input record for a *map* function and use *reduce* functions to synchronize the results produced by subgraphs.
2. We design Locality Iterative-PageRank algorithm, LI-PageRank for short, which is a novel distributed algorithm computing PageRank.

The rest of this paper is organized as follows. Section 2 presents the details of LI-PageRank algorithm. A performance analysis of our methods is presented in Section 3. We discuss related work in Section 4 and conclude the study in section 5.

2 Local Iterative PageRank Algorithm

First we look how PageRank was implemented on Hadoop and HaLoop [4] previously. As Figure 1(a) shows, L is the link table which is invariant across iterations with the format of $\langle source, destination \rangle$. R_i is the rank table at iteration i with the format of $\langle vertex, rank \rangle$. Take R_0 in Figure 1(b) for example. There are two MapReduce jobs in the loop body of naive PageRank implementation on Hadoop. The first job is to join rank table R_i and linkage table L and populate ranks; the second one is to aggregate ranks on each destination vertex. After one iteration, R_i is updated to R_{i+1} .

2.1 Locality of PageRank Computation on MapReduce

Here we extend the locality concept of PageRank computation on MapReduce by considering the computation is executed mostly in the main memory by *map* function to reduce the disc or network costs by processing messages, which are the output of *map* functions and input of *reduce* functions.

2.2 LI-PageRank Algorithm

The main idea of LI-PageRank algorithm is to reduce the communication among the Hadoop cluster. We have two advantages over previous methods. First, we need only one MapReduce job for an iteration of PageRank computation by using *map* cache to load previous PageRank values from HDFS; Secondly, we combine the in-memory PageRank computation with MapReduce by using subgraph as processing unit.

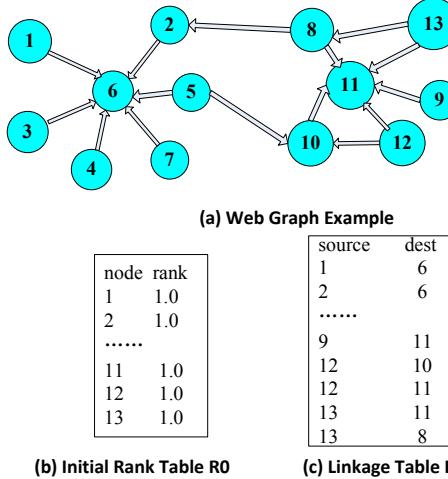


Fig. 1. Linkage Table and Initial Rank Table of Graph G

The PageRank value of vertex u at iteration i , denoted as $R_i(u)$, is dependent on the PageRank values of all vertexes pointing to vertex u where $B_i(u)$ is the set of vertexes pointing into u at partition j . Different from previous PageRank computation methods on MapReduce, we compute locally for a subgraph in a *map* function and sum up the intermediate values for each iteration. As in Equation 1, we divide the graph into k subgraphs and compute the latter part of Equation 1 by k *map* functions. Previous PageRank implementation on MapReduce processes an edge in a *map* function while our method processes a subgraph in a *map* function. Our method changes the granularity for *map* function to reduce *Shuffle Bytes*. Our method does not change the PageRank algorithm itself so the results are the same as the original one.

$$R_i(u) = \frac{(1 - d)}{n} + d * \sum_{j=1}^k \sum_{v \in B_j(u)} \frac{R_{i-1}(v)}{N_v} \quad (1)$$

For each partition, *map* function produces PageRank value for vertex v . The *mapper* (Here *mapper* means the *map* task) outputs all the partial results as \langle vertex, rank \rangle pairs to *reducer* to do the sum operation. We utilize *mapper* cache to cache the PageRank values of previous iteration by the *setup* API provided by *Hadoop*. For a subgraph, the *map* function outputs PageRank values contributed by the vertexes in the subgraph for all the vertexes, which are linked by other vertexes in this subgraph. It is much smaller comparing to previous method which outputs PageRank values as many as the number of the edges in the subgraph. In the *reduce* function, we sum up the partial results together to get the final PageRank for every vertex. There are four key steps for our method: (1) graph partitioning, (2) mapping phase as well as cache update, (3) reducing phase as well as message grouping, and (4) convergence checking.

2.3 Graph Partitioning

Our partitioning strategy is based on adjacent list of graphs. According to Equation 1 , the less common destination vertexes the subgraphs have, the less aggregation work need be done. We call a vertex as destination vertex if there is an edge pointing to it. We use the combination of horizontal partitioning and vertical partitioning to divide a graph into subgraphs. Horizontal partitioning aims to decrease the amount of PageRank values contained in mapper cache. Vertical partitioning aims to decrease the amount of output records of reducer (reduce task) output. Combination of Vertical and horizontal aim to keep the subgraphs with approximate-equal amount of edges. For the graph which has more vertexes than the threshold, horizontal partitioning is needed to divide the graph into groups. Each group contains vertexes with all their neighbors.

Map Process (PageRankMap Class)

Require: Key k, Value v
SubGraph subG=v.getSubGraph()
HashMap rankMap=subG.getRankMap
HashMap resultMap=new HashMap()
Compute PageRank for subG and store the results to resultMap
for <vertex,rank> in resultMap **do**
 Output(vertex,rank)
end for

Reduce phase (PageRankReduce Class)

Require: Key k, Set values
Output(k,Aggregate(values))

Fig. 2. LI-PageRank Implementation on Hadoop

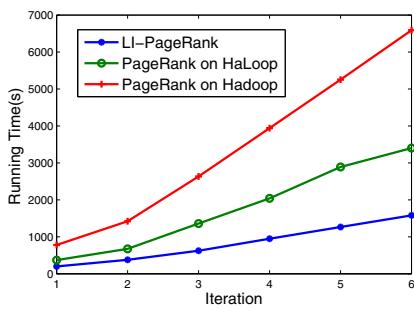
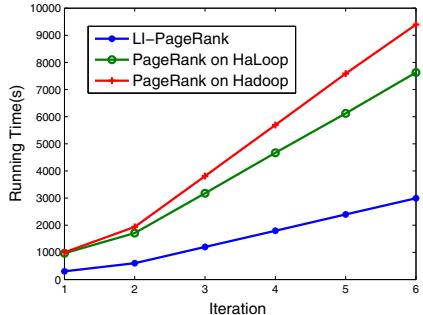
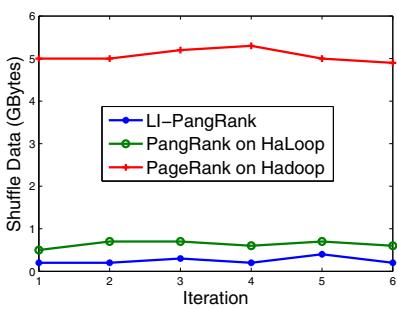
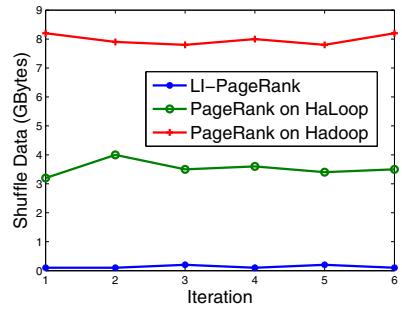
3 Experiments and Results

3.1 Dataset and Experiments Environment

We use two real-world social network datasets in this paper. The first dataset is the well-known LiveJournal dataset (4847571 vertexes, 68993773 edges). The second is Facebook dataset(957359 vertexes, 161933115 edges). These two datasets have been converted into digital format. We use a cluster with 8 working data nodes. Each node has 48TB of storage and 40GB RAM. This cluster is installed with Hadoop 0.20.2 and HaLoop.

3.2 Evaluation

We run LI-PageRank, naive PageRank on Hadoop and PageRank on HaLoop using LiveJournal dataset and Facebook dataset respectively. Both runtime and Shuffled Bytes are compared, and results are shown in Figure 3 - 6. As the figures show, for a 10-iteration job, LI-PageRank reduces the runtime to 39% compared with HaLoop and 31% compared with naive PageRank on Hadoop for facebook dataset. For LiveJournal dataset

**Fig. 3.** PageRank Performance (LiveJournal)**Fig. 4.** PageRank Performance (Facebook)**Fig. 5.** Shuffled Bytes: LiveJournal**Fig. 6.** Shuffled Bytes: Facebook

LI-PageRank reduces the runtime to 42% compared with HaLoop and 24% compared with naive PageRank on Hadoop. It can be seen that with the increase of the amount of edges, LI-PageRank is more efficient compared to other methods. Another measure is Shuffled Bytes. Shuffled Bytes means the amount of data shuffled from *map* phase to *reduce* phase. It is sorted by Hadoop according to the key comparator of the output record. Because the sort operation is quite time-consuming, we can improve the performance if we can reduce the Shuffled Bytes. HaLoop reduces the shuffled bytes of invariant table L compared to naive PageRank on Hadoop. Compared with PageRank on HaLoop, LI-PageRank reduces the shuffled bytes of both invariant table L and variant table R for *join* step. For the *aggregate* step, HaLoop can't reduce shuffled bytes because there isn't invariant table. LI-PageRank can reduce shuffled bytes by reducing the intermediate results.

4 Related Work

Link Analysis has been a popular and broadly used Web mining technique. Various link analysis schemes have been proposed. The most popular technique is Google's

PageRank [3]. PageRank is a famous algorithm and many new ranking schemes have emerged which are on basis of it. In recent years, researchers focus on parallel PageRank approaches. MapReduce computing paradigm has been a standard of distributed massive data processing platform. [6] gives a method to compute Personalized PageRank on MapReduce. [7] proposes a new approach for aggregated PageRank computation via distributed randomized algorithms. Hadoop as an open-source version of Map/Reduce [2] implementation gains its popularity by high efficiency, scalability and fault tolerance. HaLoop [4] is an extension of Hadoop, which can support iterative data processing. Twister [8] is a stream-based Map/Reduce framework which can support multi-iteration applications too. Another famous system based on Map/Reduce is HOP [9], which is a modified version of Hadoop, and allows data to be pipelined between tasks and between jobs. Google proposes Pregel [10] for large-scale graph processing. Like Hadoop, distribution-related details are hidden behind an abstract API. Pregel provides a vertex-centric approach for large-scale graph applications.

5 Conclusion and Future Work

This paper is concerned with improving the performance of the PageRank calculation using MapReduce (based on both Hadoop and HaLoop). The optimization of our method focuses on decreasing communication costs in the cluster by exploiting the locality of the PageRank algorithm for distributed system. We propose LI-PageRank algorithm to adapt hadoop to PageRank computing from a locality point of view. We will study community-based partitioning for LI-PageRank deeply in the future.

References

1. <http://hadoop.apache.org/>
2. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: OSDI (2004)
3. Page, L., Motwani, R., Brin, S., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab
4. Bu, Y., Howe, B., Balazinska, M., Ernst, M.: Haloop: Efficient iterative data processing on large clusters. PVLDB 3(1), 285–296 (2010)
5. Kang, U., Tsourakakis, C.E., Faloutsos, C.: Pegasus: A peta-scale graph mining system. In: ICDM, pp. 229–238 (2009)
6. Bahmani, B., Chakrabarti, K., Xin, D.: Fast personalized pagerank on mapreduce. In: SIGMOD Conference, pp. 973–984 (2011)
7. Ishii, H., Tempo, R., Bai, E.-W.: A new approach for aggregated pagerank computation via distributed randomized algorithms. In: CDC-ECE, pp. 6421–6426 (2011)
8. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., Fox, G.: Twister: a runtime for iterative mapreduce. In: HPDC, pp. 810–818 (2010)
9. Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Elmeleegy, K., Sears, R.: Mapreduce online. In: NSDI, pp. 313–328 (2010)
10. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: SIGMOD Conference, pp. 135–146 (2010)

A Human-Machine Method for Web Table Understanding

Guoliang Li

Department of Computer Science, Tsinghua University, Beijing, China
liguoliang@tsinghua.edu.cn

Abstract. Tabular data on the Web has become a rich source of structured data that is useful for ordinary users to explore. Due to its potential, tables on the Web have recently attracted a number of studies with the goals of understanding the semantics of those Web tables and providing effective search and exploration mechanisms over them. Table understanding is to identify, recognize and interpret tabular structures to enable a variety of tasks such as data extraction, data interpretation, data integration, and search and analysis. In this paper, we propose a human-machine hybrid method for effectively understanding tables on the Web. We develop novel techniques to address four main problems in Web table understanding: Web table extraction, Web table interpretation, Web table integration, and Web table search and analysis. We also discuss some open problems that need more research investigation in Web table understanding. We believe that Web table management will attract much more attention in the coming years.

1 Introduction

Tabular data on the Web has become a rich source of structured data that is useful for ordinary users to explore. According to a recent Google study [2], there are a total of 14 billion raw HTML tables and, among those, 150 million relational data tables, easily making Web tables one of the richest structured data sources. There are many real-world applications that can benefit from high-quality Web tables. First, search engines can utilize Web tables to improve the result quality. Existing search engines usually return relevant Web pages for queries and require users to “mine” answers from the returned Web pages. Obviously if there are many relevant high-quality Web tables, search engines can utilize these high-quality structured data to directly compute the answers and do not require users to search answers from documents. Query-answer (QA) systems can also utilize Web tables to easily generate answers for queries. Second, many systems want to provide Internet users with OLAP functionality, which is not supported in traditional keyword-based search systems. For example, if a user wants to know the average price of “iphone 5” in Beijing. Search engines cannot answer such queries. If we can extract Web tables about prices of “iphone 5” in Beijing. We can utilize these Web tables to answer such queries.

Third, knowledge bases can use Web tables to enrich themselves by adding more concepts, entities and relationships between columns.

Due to the potential, tables on the Web have recently attracted a number of studies with the goals of understanding the semantics of those Web tables and providing effective search and exploration mechanisms over them [10,5,6,2]. Table understanding is to identify, recognize and interpret tabular structures to enable a variety of tasks such as data extraction, data interpretation, data integration, and search and analysis.

There are many research challenges in Web table understanding. The first challenge is how to identify large-scale high-quality Web tables. On one hand, there are billions of Web pages and only some of them contain tabular data. On the other hand, the Web data is rather dirty (due to typing errors or different representations for the same entity). Thus it is non-trivial to identify high-quality Tables from large numbers of Web pages. The second one is how to recognize Web tables. There is usually no description information for each Web table. The column names of Web tables are also usually incomplete or inaccurate because the same concept/entity may have different representations. Thus it is rather hard to determine the subject of a Web table (the main content the Web table describes) , the concept of each column, and the entity of each cell value in the table. The third challenge is how to integrate multiple Web tables from different sources. Since Web tables from different sources may be relevant, we want to integrate them to generate more accurate and complete structured data. The fourth challenge is how to use Web tables to effectively support search and analysis applications. Internet users are only familiar with keyword queries and do not understand structured query languages. We need to bring the unstructured queries and the underlying structured data much closer so as to help users explore Web tables.

To address these challenges, we propose a human-machine hybrid method for understanding tables on the Web. We combine knowledge bases, crowds, and algorithms together and develop effective techniques to improve the quality of web table understanding. In this paper, we focus on four main problems - Web table extraction, Web table interpretation, Web table integration, and Web table search and analysis.

Paper Structure: In the paper, we first introduce our human-machine hybrid framework in Section 2 and then discuss some open problems in Section 3. Next we review some important related works in Section 4. Finally we conclude the paper in Section 5.

2 Human-Machine Hybrid Method for Web Table Understanding

2.1 Human-Machine Hybrid Framework

We propose a human-machine framework to effectively understand Web tables. Figure 2.1 shows the framework which contains four main components: Web Table

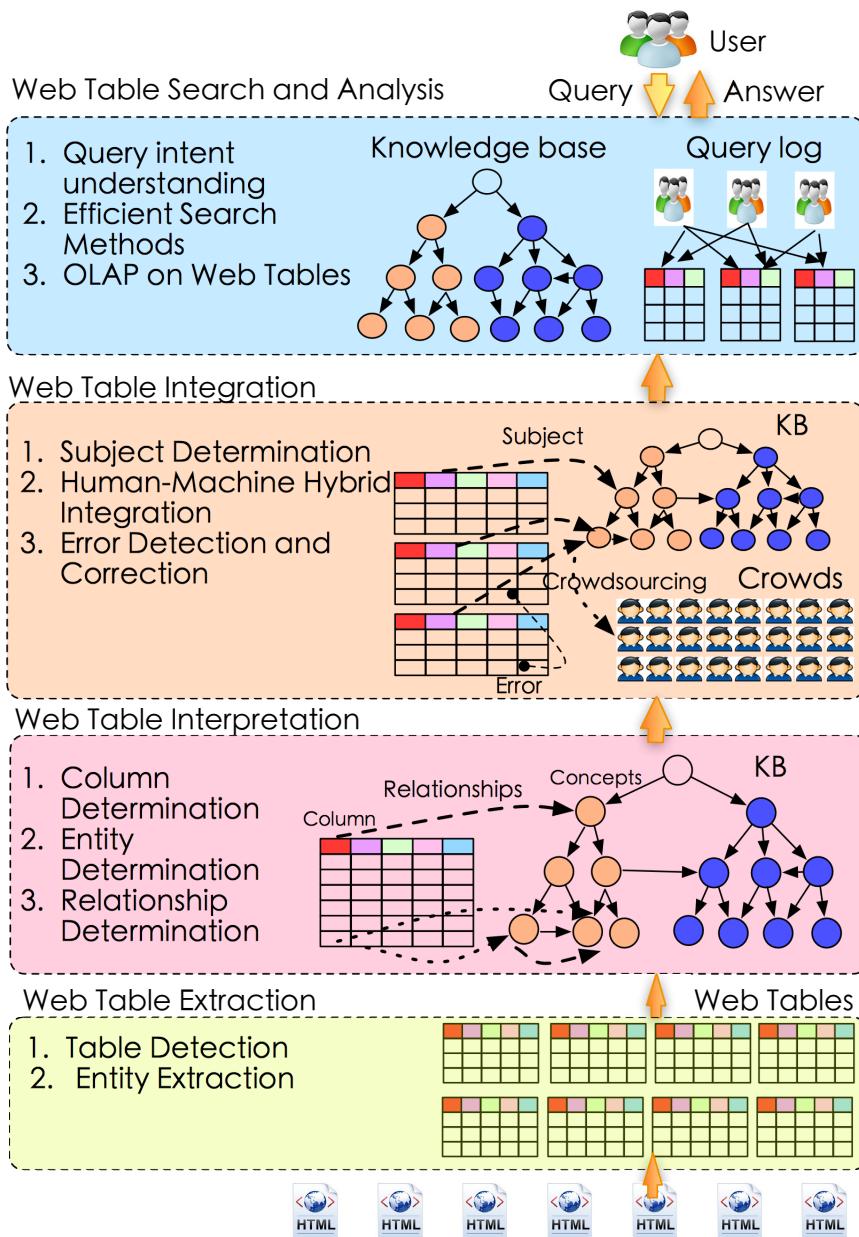


Fig. 1. Human-Machine Hybrid Framework

Extraction, Web Table Interpretation, Web Table Integration, Web Table Search and Analysis. Web Table Extraction extracts and generates Web tables from billions

of Web pages and obtains high-quality Web tables. **Web Table Interpretation** annotates Web tables as follows. Given a Web table with several columns and each column having multiple cell values, Web table interpretation wants to (1) label each column with top- k relevant concepts; (2) label each cell value in the table with top- k entities; (3) identify the relationships between columns; and (4) find the primary keys of the Web table. **Web Table Integration** integrates multiple Web tables by finding correspondences between their primary keys and other columns. In other words, it finds the table pairs with primary keys corresponding to the same concept and correlates other columns which also correspond to the same concept. **Web table integration** has the following differences with traditional schema mapping. First, the tables in the schema mapping problem are usually well-structured and clean while Web tables are rather dirty (Since there may exist errors and inconsistencies in Web data). Second, the primary keys in Web tables are not given. Third, the number of cell values in Web tables is usually much smaller than that in schema-matching problems. Thus traditional instance-level schema mapping techniques may not work for integrating Web tables. **Web Table Search and Analysis** provides users with search and analysis functionalities. For example, users can search on the Web tables and get analysis results from We tables. Next we discuss the details of each component.

2.2 Web Table Extraction

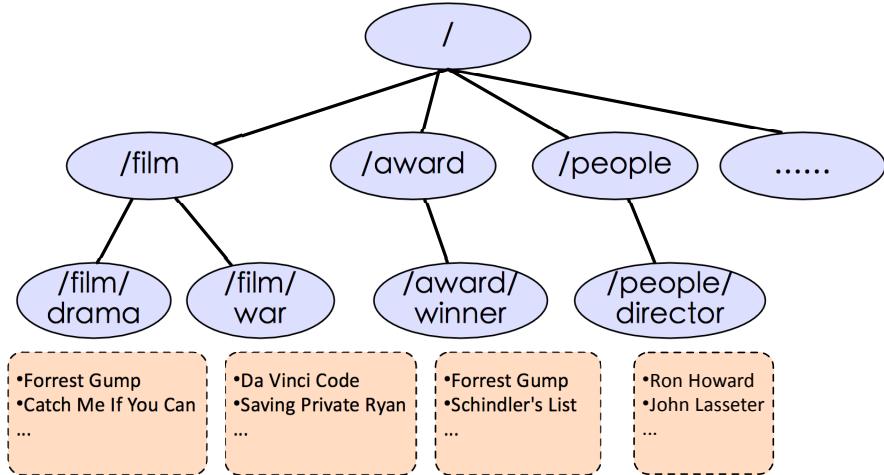
To extract Web tables from Web pages, we enumerate each Web page and detect whether the Web page contains tables. If yes, we extract the tables from the page as follows. We use the rule-based method to detect and extract the Web tables. The basic idea is that if a page contains Web tables, it usually contains some important keywords, e.g., “table, tr, th, td”. We can utilize these rules to extract Web tables. In addition, to extract Web tables in a specific topic, we employ a dictionary based entity extraction method. We take the entities in the given topic as a dictionary and check whether the Web page contains enough entities in the dictionary. We can also support approximate matching between entities and substrings in a dictionary by using similarity functions (e.g., Jaccard and Edit distance) to quantify the similarity. Then we can extend approximate entity extraction methods to address the Web table extraction problem. Interested readers refer to [12,3] for more details about approximate directionally based entity extraction.

2.3 Web Table Interpretation

We utilize knowledge bases to effectively annotate Web tables. A knowledge base consists of many concepts and each concept has large numbers of entities. Different concepts may have “type-subtype” relationships. For example, Figure 2 shows a part of a knowledge base. In the figure, the leaf nodes are entity set and intermediate nodes are concepts.

For each column, we identify the concepts that have large similarity with the column to annotate the column. The similarity between a column and a concept

1994	Forrest Gump	Robert Zemeckis	Tom Hanks, Robin Wright, Sally Field,...
1998	Saving Private Ryan	Steven Spielberg	Tom Hanks, Tom Sizemore, Matt Damon,...
2002	Catch Me If You Can	Steven Spielberg	Tom Hanks, Christopher Walken,...
2006	The Da Vinci Code	Ron Howard	Tom Hanks, Ian McKellen, Jean Reno,...

Fig. 2. An example Web table**Fig. 3.** An example knowledge base

can be quantified by using existing set similarity functions (on the set of cell values of the column and the set of entities of the concept), e.g., Jaccard coefficient or Cosine similarity. For example, the Jaccard similarity between the second column of the Web table to the concepts “/film”, “/film/drama”, “/film/war” are respectively 3/5, 1/2, and 1/5. If we want to select top-1 concept for the second column, it should be “/film”.

There are several challenges to annotate the Web tables using knowledge bases. First, there are large numbers of Web tables, and knowledge bases have large numbers of entities. It is challenging to annotate Web tables using knowledge bases, especially finding top- k concepts for each column. To address this challenge, we devise a Map-Reduce based similarity join method. We extend the partition-based method to support similarity joins in parallel [13]. Second, since Web tables many contain errors or the same entity may have different representations, it is very important to support fuzzy matching between cell values and entities. For example, although the cell value “The Da Vinci Code” in the Web table does not exactly match the entity “Da Vinci Code” in the knowledge base, they should represent the same entity. If we can support fuzzy matching, they can be approximately matched and identified as the same entity and thus the

fuzzy matching based method can improve the annotation quality. To address the second challenge, we use hybrid similarity functions which tolerate fuzzy matching between tokens in set similarity functions. Interested readers refer to [27] for details about the hybrid similarity functions and how to support such complex similarity functions.

After annotating the columns, we can get top- k concepts of each column. Then for each cell value, we can find top- k entities from the entities of these concepts. Notice that this problem can be reduced to the problem of top- k similarity join, which, given two sets of entities, for each entity in a set, finds top- k similar entities from another set. We devise efficient similarity join based algorithms to address this problem [4,13,26].

In terms of relationship discovery between different columns, we can model the problem as the Steiner graph problem which finds Steiner trees from a graph. The concepts in the knowledge base can be modelled as a graph where nodes are concepts and edges are relationships between concepts. Given a Web table, each column corresponds to several nodes (entities) in the knowledge base. We want to find a Steiner tree which covers all columns with the minimum cost (i.e., the sum of the edge weights in the Steiner tree). We adopt existing Steiner tree based method to address this problem [17].

2.4 Web Table Integration

Web table integration includes three steps. In the first step, we require to discover the primary keys (or subjects) of each Web table. In the second step, we identify the concepts of each column. In the third step, we integrate Web tables with their primary keys mapping to the same concept. These Web tables can be integrated together. Then we integrate other columns which map to the same concept. Next we discuss the details of each step.

Primary Key Discovery: We utilize the Steiner tree of each Web table to discover its primary keys. Intuitively, the column that has (directly or indirectly) relationships to each other column should be the primary keys. We can use this idea to identify the primary keys. To improve the quality, we propose a human-machine hybrid method. We first cluster the Steiner trees based their structures. Then from each cluster, we select some Steiner trees as representatives and ask crowds for finding the accurate primary keys of these Steiner trees. Based on the results of representatives, we utilize them to select primary keys for other Steiner trees with similar structures. In this way, we can utilize crowds to improve the quality.

Human-Machine Concept Determination: In the second step, we identify the concepts of each column. Although we can use the concept determination method discussed in Section 2.3, the quality may be not good enough. To address this problem, we propose a human-machine based concept determination method. First, we use the machine based method to identify top- k concepts of each column. Then we construct a graph where nodes are columns and concepts. There is an edge from a column to a concept/column if their similarity

is not smaller than a threshold and the edge weight is their similarity. Next we want to select some edges to ask questions from crowds and utilize the answers to judge whether we need to check other edges. Notice that we can utilize the “transitivity” property to reduce the number of questions. The basic idea is that given two concepts c_1 and c_2 and a concept T . If the similarity between c_1 and T is very large, and the similarity between c_1 and c_2 is also very large, then we can deduce that the similarity between c_2 and T should be large based on the transitivity rule. In this way, we only need to ask the question between c_1 and T and avoid the question between c_2 and T . Thus we can reduce the money cost. This is a challenge to select “important” edges to ask questions and how to ask the questions “in an appropriate order”. We have proved that it is better to first ask questions of edges with large similarity [28]. Based on this idea, we can devise efficient methods to select edges to ask questions.

Web Table Integration: We identify the tables whose primary keys correspond to the same concept and integrate them together as follows. For two tables that can be integrated on primary keys, we check whether their other columns can map to the same concept. If yes, we also integrate these columns. In this way, we can integrate multiple Web tables.

2.5 Web Table Search and Analysis

Given the Web tables, we need to provide the Internet users with search and analysis functionalities. There are two challenges. First, how to judge whether a query can be effectively answered by Web tables. Second, how to efficiently answer search and aggregation queries on Web tables. To address the first challenge, we use query logs to help us check whether Web tables contain the answers of a query. We construct a bipartite graph where nodes are queries and Web tables and there is an edge between a query and a Web table if the Web table is an answer of the query. Based on the bipartite graph, for each new query, we can determine whether we can use Web tables to answer the query. For the second challenge, there are three main search strategies: keyword search, form-based search (aka. query by example - QBE), and SQL. However they have some limitations. First, keyword search does not support aggregation queries. Second, SQL is hard to use for Internet users since it requires users to know SQL syntax and the underlying schema. Third, the form based method has less usability than keyword search and worse expressiveness than SQL. To address this problem, we propose a new search paradigm, called search-as-you-type, to improve user search experiences.

Search-as-You-Type: It returns answers as users type in queries letter by letter. It can provide users with instant feedback and help users to modify their input queries. It can reduce users typing effort. Moreover, our method can also tolerate the errors and inconsistencies between queries and the underlying data. We utilize similarity functions to tolerate the errors, e.g., Edit Distance. In other words, even if the data approximately matches the query, we still can find it as an answer. The big challenge is to support instant feedback for each keystroke,

which is usually in several milliseconds. To address this issue, we devise effective trie-based indexing structures and efficient search algorithms [18,15,11,16,8].

Search-as-You-Type on Forms: Keyword search methods cannot support aggregation queries. To address this limitation, we enable search-as-you-type on forms [14]. As users type in queries letter by letter in forms, we can on-the-fly return answers and aggregate on the attribute the users are typing in. The big challenge is how to support on-the-fly aggregation and search on multiple attributes. We devise effective multi-trie based index and search algorithms [14].

SQL Suggestion: In many real-world applications, many users must type in SQL queries, e.g., Database Administrator (DBA), SQL programmers. To boost users SQL coding productivity, we propose SQL suggestion which on-the-fly suggests SQL queries as users type in keywords letter by letter [7]. The big challenge is how to generate structured queries based on limited keywords. Moreover, we want to support aggregation queries, e.g., max, count, min. It is rather challenging to support structure queries as well as aggregation queries. We propose template-based index structure and threshold-based algorithms to effectively generate SQL queries [7].

3 Open Problems

There are still many open problems and research opportunities in Web table understanding.

3.1 Extracting Web Tables from Unstructured Data

The number of tables on the Web is limited and large volume of Web content is in unstructured format. Thus Web tables are not enough to support various applications. If we can extract more Web tables from unstructured data, we can increase the number of Web tables so as to cover more real-world applications. The challenge is how to extract and integrate structured data from unstructured content. A possible way is to use knowledge bases to guide the extraction and integration. It calls for effective techniques to support semantic and structure aware extraction and integration.

3.2 Quality Control in Web Table Interpretation

Many cell values in Web tables may not appear in knowledge bases, and in this way the knowledge base interpretation method cannot understand such tables. Although we can utilize crowds to annotate these tables, crowds may return inaccurate answers for such “uncommon” data (since they do not appear in knowledge bases, they should be infrequent and thus crowds may not be familiar with such uncommon data), thus the quality cannot be guaranteed. In addition, workers in crowdsourcing platforms may also return inaccurate answers, we need to devise effective methods to tolerate the errors. To address these problems, it calls for effective quality control techniques to guarantee the quality in Web table interpretation.

3.3 Large-Scale Web Table Integration

Due to the high-money-cost and large-latency limitations in crowdsourcing, it is rather expensive and time consuming to use crowds to annotate and integrate Web tables. It calls for new methods to support large numbers of Web tables. It may be an opportunity to combine machine learning algorithms, knowledge bases, and crowds to integrate multiple Web tables. First, the crowds only annotate some Web tables that are hard to correctly annotate for algorithms. Then, we use machine leaning based algorithms to use crowd's answers to annotate other Web tables so as to reduce the number of questions submitted to crowds. Iteratively, we can integrate large numbers of Web tables with less money cost and high quality.

4 Related Works

In this section, we review some important existing studies that are related to Web table understanding.

Web Table Systems: Google has launched a FusionTable project [10] to gather high-quality relational tables from the Web. Elmeleegy et al. [5,6] proposed to extract and integrate tables from Web lists. Cafarella et al. [2] built a WebTable system by extracting and leveraging the relational information embedded in HTML tables on the Web.

Web Table Annotation: Limaye et al.[19] studied how to annotate columns of Web tables, i.e., determining the concept of each column of Web tables, the entity of each cell value, and the relationship between columns. They employed machine learning based techniques which are neither scalable nor efficient. Venetis et al. proposed a simple majority-rule mechanism of selecting the concept(s) that the most cells in the column have been mapped to, but with a much larger knowledge base that is derived from the Web [25]. Obviously this method does not support fuzzy matching between cell values and entities and leads to low recall. Yakout et al. [30] studied how to augment entities with attribute values and discovering attributes using Web tables. Pimplikar et al. [23] studied how to answer table queries based on column keywords.

Knowledge Bases: Freebase [1] is a collaboratively created graph database for structuring human knowledge which is manually built by thousands of data-lovers. Freebase contained about 4 thousand types and 40 million entities. Probase [29] is a universal, probabilistic taxonomy which is harnessed from billions of Web pages using some predefined patterns, such as “isA” and “such as” patterns. Probase included more than 2.7 million types and about 22 million entities. Yago [24] is a large semantic knowledge base which is derived from Wikipedia, WordNet and GeoNames. Yago contained about 0.3 million types and 9 million entities.

Crowdsourcing: Crowdsourcing recently attracted significant attention from both industrial and academic communities. Amazon developed a crowdsourcing platform, called Amazon Mechanical Turk (<https://www.mturk.com/mturk/>

welcome). There are about millions of active users and we can conduct real experiments on this platform. From database communities, Franklin et al. [9] developed a crowdsourcing database system to answer database-hard queries. Parameswaran et al. developed [22] efficient filtering algorithms using crowds. Liu et al. [20] developed a crowdsourcing data analytics system. Marcus et al.[21] enabled crowds to support joins and sorts. We studied the crowdsourcing entity solution problem [28].

5 Conclusion

In this paper, we introduce the potential of Web tables and discuss the research challenges in Web table understanding. We propose a human-machine framework for effectively and efficiently understanding Web tables. We develop effective indexes and algorithms to address four main problems in Web table understanding: Web table extraction, Web Table Interpretation, Web table integration, and Web table search and analysis. We also discuss some open problems and research opportunities in Web table management. We believe that Web table management will attract much more attention from both industrial and academic communities in the near future.

Acknowledgement. This work was partly supported by the National Natural Science Foundation of China under Grant No. 61003004 and 61272090, National Grand Fundamental Research 973 Program of China under Grant No. 2011CB302206, and a project of Tsinghua University under Grant No. 20111081073, and the “NExT Research Center” funded by MDA, Singapore, under Grant No. WBS:R-252-300-001-490.

References

1. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: A collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
2. Cafarella, M.J., Halevy, A.Y., Wang, D.Z., Wu, E., Zhang, Y.: Webtables: exploring the power of tables on the web. VLDB 1(1), 538–549 (2008)
3. Deng, D., Li, G., Feng, J.: An efficient trie-based method for approximate entity extraction with edit-distance constraints. In: ICDE, pp. 762–773 (2012)
4. Deng, D., Li, G., Feng, J.: Top-k string similarity search with edit-distance constraints. In: ICDE (2013)
5. Elmeleegy, H., Madhavan, J., Halevy, A.Y.: Harvesting relational tables from lists on the web. VLDB 2(1), 1078–1089 (2009)
6. Elmeleegy, H., Madhavan, J., Halevy, A.Y.: Harvesting relational tables from lists on the web. VLDB J. 20(2), 209–226 (2011)
7. Fan, J., Li, G., Zhou, L.: Interactive sql query suggestion: Making databases user-friendly. In: ICDE, pp. 351–362 (2011)
8. Feng, J., Li, G.: Efficient fuzzy type-ahead search in xml data. IEEE Trans. Knowl. Data Eng. 24(5), 882–895 (2012)

9. Franklin, M.J., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: Crowddb: Answering queries with crowdsourcing. In: SIGMOD Conference, pp. 61–72 (2011)
10. Gonzalez, H., Halevy, A.Y., Jensen, C.S., Langen, A., Madhavan, J., Shapley, R., Shen, W., Goldberg-Kidon, J.: Google fusion tables: Web-centered data management and collaboration. In: SIGMOD Conference, pp. 1061–1066 (2010)
11. Ji, S., Li, G., Li, C., Feng, J.: Efficient interactive fuzzy keyword search. In: WWW, pp. 371–380 (2009)
12. Li, G., Deng, D., Feng, J.: Faerie: Efficient filtering algorithms for approximate dictionary-based entity extraction. In: SIGMOD Conference, pp. 529–540 (2011)
13. Li, G., Deng, D., Wang, J., Feng, J.: Pass-join: A partition-based method for similarity joins. VLDB 5(3), 253–264 (2011)
14. Li, G., Fan, J., Wu, H., Wang, J., Feng, J.: Dbease: Making databases user-friendly and easily accessible. In: CIDR, pp. 45–56 (2011)
15. Li, G., Ji, S., Li, C., Feng, J.: Efficient type-ahead search on relational data: a tastier approach. In: SIGMOD Conference, pp. 695–706 (2009)
16. Li, G., Ji, S., Li, C., Feng, J.: Efficient fuzzy full-text type-ahead search. VLDB J. 20(4), 617–640 (2011)
17. Li, G., Ooi, B.C., Feng, J., Wang, J., Zhou, L.: Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In: SIGMOD Conference, pp. 903–914 (2008)
18. Li, G., Wang, J., Li, C., Feng, J.: Supporting efficient top-k queries in type-ahead search. In: SIGIR, pp. 355–364 (2012)
19. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. VLDB 3(1), 1338–1347 (2010)
20. Liu, X., Lu, M., Ooi, B.C., Shen, Y., Wu, S., Zhang, M.: Cdas: A crowdsourcing data analytics system. VLDB 5(10), 1040–1051 (2012)
21. Marcus, A., Wu, E., Karger, D.R., Madden, S., Miller, R.C.: Human-powered sorts and joins. VLDB 5(1), 13–24 (2011)
22. Parameswaran, A.G., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A., Widom, J.: Crowdscreen: algorithms for filtering data with humans. In: SIGMOD Conference, pp. 361–372 (2012)
23. Pimplikar, R., Sarawagi, S.: Answering table queries on the web using column keywords. VLDB 5(10), 908–919 (2012)
24. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A core of semantic knowledge. In: WWW, pp. 697–706 (2007)
25. Venetis, P., Halevy, A.Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. VLDB 4(9), 528–538 (2011)
26. Wang, J., Li, G., Feng, J.: Trie-join: Efficient trie-based string similarity joins with edit-distance constraints. PVLDB 3(1), 1219–1230 (2010)
27. Wang, J., Li, G., Feng, J.: Fast-join: An efficient method for fuzzy token matching based string similarity join. In: ICDE, pp. 458–469 (2011)
28. Wang, J., Li, G., Kraska, T., Franklin, M.J., Feng, J.: Leveraging transitive relations for crowdsourced joins. In: SIGMOD (2013)
29. Wu, W., Li, H., Wang, H., Zhu, K.Q.: Probbase: a probabilistic taxonomy for text understanding. In: SIGMOD Conference, pp. 481–492 (2012)
30. Yakout, M., Ganjam, K., Chakrabarti, K., Chaudhuri, S.: Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In: SIGMOD Conference, pp. 97–108 (2012)

Probabilistic k -Skyband Operator over Sliding Windows^{*}

Xing Feng¹, Wenjie Zhang², Xiang Zhao², Ying Zhang², and Yunjun Gao¹

¹ Zhejiang University

² University of New South Wales

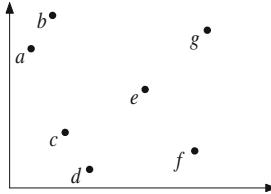
{xingfeng, zhangw, xzhao, yingz}@cse.unsw.edu.au, gaojy@zju.edu.cn

Abstract. Given a set of data elements \mathcal{D} in a d -dimensional space, a k -skyband query reports the set of elements which are dominated by at most $k - 1$ other elements in \mathcal{D} . k -skyband query is a fundamental query type in data analyzing as it keeps a minimum candidate set for all top- k ranking queries where the ranking functions are monotonic. In this paper, we study the problem of k -skyband over uncertain data streams following the possible world semantics where each data element is associated with an occurrence probability. Firstly, a dynamic programming based algorithm is proposed to identify k -skyband results for a given set of uncertain elements regarding a pre-specified probability threshold. Secondly, we characterize the minimum set of elements to be kept in the sliding window to guarantee correct computing of k -skyband. Thirdly, efficient update techniques based on R-tree structures are developed to handle frequent updates of the elements over the sliding window. Extensive empirical studies demonstrate the efficiency and effectiveness of our techniques.

1 Introduction

In a d dimensional numerical space, we say an element e dominates another element e' if e is not worse than e' in every dimension and is better than e' in at least one dimension. Given a dataset \mathcal{D} , the k -skyband of \mathcal{D} contains the set of points which are dominated by at most $k - 1$ other points. k -skyband is a useful tool in data analyzing as it retrieves the objects which are definitely worse than (i.e., dominated by) at most $k - 1$ other objects. The concept of k -skyband is also closely related to the following two concepts, *skyline* and *top- k ranking* queries. Given a dataset \mathcal{D} , the *skyline* of \mathcal{D} contains all elements which are not dominated by any other elements from \mathcal{D} . Clearly, skyline is a special case of k -skyband query where $k = 1$. Given a dataset \mathcal{D} and a ranking function f , a top- k query retrieves the k elements in \mathcal{D} with the highest scores according to f . An important property of k -skyband is that it keeps a minimum candidate set to answer multiple ad-hoc top- k queries given that the ranking functions of these queries are monotonic. Thus, keeping k -skyband of a dataset is sufficient to answer all top- k

* Wenjie Zhang was partially supported by ARC DE120102144 and DP120104168. Ying Zhang was partially supported by DP110104880 and UNSW ECR grant PSE1799. Yunjun Gao was supported in part by NSFC 61003049 and the Fundamental Research Funds for the Central Universities under Grant 2012QNA5018.

**Fig. 1.** Skyband Example

ranking queries with monotonic ranking functions. In Figure 1, the skyline elements (1-skyband) are $\{a, c, d\}$ as neither of them is dominated by other elements; 2-skyband consists $\{a, c, d, b, f\}$ which are dominated by at most 1 other object. Note that without loss of generality, we assume smaller values are preferred in the paper.

Uncertain data analysis is an important issue in many emerging important applications, such as sensor networks, trend prediction, moving object management, data cleaning and integration, economic decision making, and market surveillance. Uncertainty arises in these applications due to various reasons including data randomness and incompleteness, delay or loss in data transfer, limitation of equipments, privacy preservation, etc. In many scenarios of such applications, uncertain data is collected in a streaming fashion. Uncertain streaming data computation has been studied and the existing work mainly focuses on aggregate queries, top- k queries, skyline queries, etc [3,6,15,16]. In this paper, we will investigate the problem of efficient skyband computation over uncertain streaming data where each data element has a probability to occur.

Skyband computation over uncertain streaming data has many applications. For instance, in an on-line shopping system products are evaluated in various aspects such as *price* and *condition*. In addition, each seller is associated with a “trustability” value which is derived from customers’ feedback on the seller’s product quality, delivery handling, etc. This “trustability” value can also be regarded as occurrence probability of the product since it represents the probability that the product occurs exactly as described in the advertisement in terms of delivery and quality. A customer may want to select a product, say laptops, according to multi-criteria based ranking, such as low price, good condition, etc, and require that the retrieved laptop cannot be definitely worse than (i.e., dominated by) $k - 1$ other laptops where k is specified by the user. In Table 1, L_1 and L_4 are skyline points, $\{L_1, L_4, L_2, L_3\}$ form 2-skybands. Nevertheless, we should also consider that L_1 is posted long time ago; L_4 is better than (dominates) L_3 but the trustability of the seller of L_4 is low.

Table 1. Laptop Advertisements

Product ID	Time	Price	Condition	Trustability
L_1	107 days ago	\$ 550	excellent	0.80
L_2	5 days ago	\$ 680	excellent	0.90
L_3	2 days ago	\$ 530	good	1.00
L_4	today	\$ 200	good	0.48

The challenges for skyband computation over uncertain sliding windows are three-fold. Firstly, computing probabilistic k -skyband for a given set of uncertain elements is costly since naively we need to enumerate all the possible worlds which is exponential to the number of total elements. Secondly, a critical requirement in data stream computation is to have small space so it is important to identify a candidate set with minimum size for correct k -skyband computation in the sliding window. Thirdly, due to high arriving speed and high data volume, it is essential to develop time efficient techniques to handle frequent updates (i.e., insertions and deletions) in the sliding window.

To the best of our knowledge, this paper is the first to study the problem of probabilistic k -skyband operator over uncertain sliding windows. Our main contributions are summarized as follows.

- We formally define the probabilistic k -skyband operator in uncertain sliding windows following the possible world semantics in a probability threshold fashion.
- A dynamic programming based algorithm is proposed to efficiently retrieve k -skyband elements for a given set of uncertain elements. The algorithm runs in $O(nk)$ time for an element a where n is the number of elements dominating a . An optimization technique is further proposed to reduce the time complexity to $O(k)$ when a new element is added to the dominating element set of a .
- We characterize the minimum information needed in continuously computing k -skyband regarding a sliding window and show that the expected size of the minimum candidate set is $k \ln^d \frac{N}{k}$ when the data set follows uniform distribution.
- We develop novel, efficient, incremental techniques to continuously compute probabilistic k -skyband over sliding windows.
- Space and time efficiency are demonstrated by an extensive empirical study on both real and synthetic datasets.

The rest of the paper is organized as follows. In Section 2, we formally define the problem of sliding-window k -skyband computation and provide some necessary background information. Section 3 presents the theoretical foundations. We develop techniques for k -skyband probabilities computation in Section 4 and continuous updates of probabilistic k -skyband queries in Section 5. Results of comprehensive performance evaluation are presented in Section 6. Section 7 summarizes related work and Section 8 concludes the paper.

2 Background Information

We use DS to represent a sequence (stream) of data elements in a d -dimensional numeric space such that each element a has a probability $P(a)$ ($0 < P(a) \leq 1$) to occur where $a.i$ (for $1 \leq i \leq d$) denotes the i -th dimension value. For two elements u and v , u dominates v , denoted by $u \prec v$, if $u.i \leq v.i$ for every $1 \leq i \leq d$, and there exists a dimension j with $u.j < v.j$. Given a set of elements, the k -skyband consists of all points which are dominated by at most $k - 1$ other elements.

2.1 Problem Definition

Given a sequence DS of uncertain data elements, a *possible world* W is a subsequence of DS . The probability of W to appear is $P(W) = \prod_{a \in W} P(a) \times \prod_{a \notin W} (1 - P(a))$. Let

Ω be the set of all possible worlds, then $\sum_{W \in \Omega} P(W) = 1$. We use $SKYBAND(W)$ to denote the set of elements in W that form the k -skyband of W . The probability that an element a appears in the k -skyband of the possible worlds is $P_{skyband}(a) = \sum_{a \in SKYBAND(W), W \in \Omega} P(W)$. $P_{skyband}(a)$ is called the k -skyband probability of a .

In many applications, a data stream DS is *append-only* [7,9,14]; that is, there is no deletion of data element involved. In this paper, we study the k -skyband computation problem restricted to the append-only data stream model. In a data stream, elements are positioned according to their relative arrival ordering and labelled by integers. Note that the position/label $\kappa(a)$ means that the element a arrives $\kappa(a)$ th in the data stream.

Problem Statement. In this paper, we study the problem of efficiently retrieving k -skyband elements from the most recent N elements, seen so far, with the k -skyband probabilities not smaller than a given probability threshold q ($0 < q \leq 1$).

2.2 Preliminaries

An Assisted Probability. Let DS_N denote the most recent N tuples. For each element a , we use $P_{new}(a)$ to denote the k -skyband probability of a restricted to the elements newer than a only.

Example 1. Suppose 7 uncertain elements a, b, c, d, e, f, g arrive according to alphabetic order. For an element c , $P_{skyband}(c)$ denote the probability that c is dominated by at most $k - 1$ elements among a, b, d, e, f, g . On the other hand, $P_{new}(c)$ is the probability that c is dominated by at most $k - 1$ elements among d, e, f, g (i.e., newer than c) only.

3 Framework

Given a probability threshold q and a sliding window with length N , Algorithm 1 shows the framework for continuously processing probabilistic k -skyband queries where a_{old} is the oldest element in the current window DS_N . Functions $insert(a_{new})$ and $remove(a_{old})$ will be elaborated in the following sections.

Algorithm 1. Probabilistic k -skyband Computation over a Sliding Window

```

1 while a new element  $a_{new}$  arrives do
2   if  $|S| \leq N$  then
3     insert( $a_{new}$ );
4   else
5     remove( $a_{old}$ );
6   insert( $a_{new}$ );

```

3.1 Using $S_{N,q}$ only

Let $S_{N,q}$ denote the set of elements from DS_N with their P_{new} values not smaller than q ; that is,

$$S_{N,q} = \{\alpha | \alpha \in DS_N \& P_{new}(\alpha) \geq q\} \quad (1)$$

A critical requirement in data stream computation is to have small memory space and fast computation. In our algorithms, instead of keeping all recent N elements DS_N , we propose to keep only $S_{N,q}$ which is shown to be logarithmic in size regarding N in the empirical study. Next, we show the correctness of using $S_{N,q}$ only in the computation.

Theorem 1. $S_{N,q}$ is the minimum set of elements to be maintained to correctly compute probabilistic k -skyband queries for sliding window size of N regarding probability threshold q .

The proofs of Theorem 1 is lengthy and we omit it here due to space limits. The correctness of the theorem is based on the following properties of $S_{N,q}$.

- Processing k -skyband query based on $S_{N,q}$ only will not miss any k -skyband points.
- For a k -skyband element, its k -skyband probability computed based on $S_{N,q}$ only is equal to that computed based on all most recent N elements.
- $S_{N,q}$ is the minimum set of elements to guarantee correct retrieval of k -skyband results within the most recent N elements.

Size of $S_{N,q}$. Elements in the candidate set $S_{N,q}$ can be regarded as the k -skyband results in the $d + 1$ dimensional space by including *time* as an additional dimension. Thus, with the assumption that all points follow the uniform distribution, the expected size of $S_{N,q}$ is $k \ln^d \frac{N}{k}$.

4 k -Skyband Probability Calculation

For a given uncertain element a in the sliding window, assume that a is dominated by n other elements u_1, \dots, u_n . Note that u_1, \dots, u_n could be efficiently retrieved using any existing dominance reporting algorithms based on R-trees [9]. Clearly, the k -skyband probability of a is the sum of probabilities of all possible worlds in which a is dominated by at most $k - 1$ other elements. Namely, at most $k - 1$ elements from u_1, \dots, u_n appear since the occurrence of elements not dominating a does not affect the computation of $P_{skyband}(a)$.

Next we present a dynamic programming based algorithm for calculating $P_{skyband}(a)$. Assume that the elements u_1, \dots, u_n are sorted according to an arbitrary order, we build a matrix M with size $(n + 1) \times (n + 1)$ as follows. Let $m_{i,j}$ denote the probability that exactly i elements out of the first j elements happen. The following equation is immediate.

$$m_{i,j} = m_{i-1,j-1} \times P(u_j) + m_{i,j-1} \times (1 - P(u_j)) \quad (2)$$

where $P(u_j)$ is the occurrence probability of element u_j . In the initializing phase, $m_{0,j}$ (for $1 \leq j \leq n$) equals $(1 - P(u_1)) \times \dots \times (1 - P(u_j))$, namely none of the first i elements occurs; $m_{i,i}$ (for $1 \leq i \leq n$) equals $P(u_1) \times \dots \times P(u_i)$ meaning that all first i elements occur. The construction of the matrix is summarized in the following equation.

$$m_{i,j} = \begin{cases} (1 - P(u_1)) \times \dots \times (1 - P(u_i)) & j = 0 \\ P(u_1) \times \dots \times P(u_i) & i = j \\ m_{i-1,j-1} \times P(u_j) + m_{i,j-1} \times (1 - P(u_j)) & otherwise \end{cases}$$

Algorithm 2. Skyband Probability Calculation

```

Input :  $a, u_1, \dots, u_n, k, q$ 
Output :  $P_{skyband}(a)$ 
1 if  $n \leq k - 1$  then
2   return 1;
3  $m_{0,0} = 1;$ 
4 for  $1 \leq i \leq n$  do
5    $m_{0,i} = m_{0,i-1} \times (1 - P(u_i));$ 
6    $m_{i,i} = m_{i-1,i-1} \times P(u_i);$ 
7 for  $1 \leq i \leq k - 1$  do
8    $j = i + 1;$ 
9   while  $j \leq n$  do
10     $m_{i,j} = m_{i-1,j-1} \times P(j) + m_{i,j-1} \times (1 - P(j));$ 
11 return  $\sum_{i=0}^{k-1} m_{i,n};$ 

```

The algorithm for computing $P_{skyband}(a)$ is reported in Algorithm 2.

Time Complexity. In building the matrix, the sum of values $m_{i,n}$ ($0 \leq i \leq k - 1$) captures the probability that at most $k - 1$ elements occur among the n elements dominating a . Thus, we can stop once $m_{k,n}$ is retrieved, leading to the time complexity $O(kn)$.

Optimization. Since the elements dominating a are sorted in an arbitrary order, if a new element is inserted into the dominating set of a , we do not need to rebuild the whole matrix. Instead, the existing $n \times n$ matrix can be expanded to an $(n+1) \times (n+1)$ matrix and we retrieve the probability that at most k elements occur out of $n+1$ elements from the new matrix. The time complexity in this case is reduced to $O(k)$.

5 Continuously Processing Probabilistic k -Skyband Queries

A naive execution of Algorithm 1 is with every update of the sliding window (i.e., arrival of a_{new} and expiration of a_{old}), for every element a in $S_{N,q}$, we recompute its k -skyband probability $P_{skyband}(a)$ following Algorithm 2 regarding the updates, and report elements with $P_{skyband}(a) \geq q$. In this section, we present novel aggregate information enhanced, R-tree based algorithms to efficiently handle the updates.

5.1 Aggregate R-Tree

We monitor the following two probability values in the continuous processing, P_{new} and $P_{skyband}$. Once $P_{new}(a)$ for an element a falls below the probability threshold q , as shown in Section 3, a could be safely removed. $P_{skyband}(a)$ for an element a decides its qualification as a k -skyband result. Specifically, we continuously keep two R-trees, R_1 indexes all candidate elements in $S_{N,q}$ which are not the k -skyband results in the sliding window, where R_2 indexes all current k -skyband results. Clearly, R_1 and R_2 together keep all candidate elements from $S_{N,q}$.

Furthermore, for each element a in R_1 and R_2 . Denote the matrix to compute $P_{skyband}(a)$ in Section 4 as $M_{skyband}$. We maintain the last column of $M_{skyband}$, denoted as $\lambda_{M_{skyband}}(a)$ so that when a new element is added to the element list dominating a , $P_{skyband}$ can be updated by using $\lambda_{M_{skyband}}(a)$ only in $O(k)$ time. Similarly, denote the matrix for calculating $P_{new}(a)$ as M_{new} . Its last column $\lambda_{M_{new}}(a)$ is kept to update $P_{new}(a)$ in $O(k)$ time once a new element is added to the dominance list.

5.2 Handling Insertion of a_{new}

When a new tuple a_{new} arrives in the sliding window, it stays in $S_{N,q}$ because it is not dominated by any element newer than it and thus $P_{new}(a_{new}) = 1$. After computing the $P_{skyband}$ value of a_{new} , we may decide if it is a candidate element (to insert into R_1) or a k -skyband result (to insert into R_2). Besides, the following effects are invoked by the arrival of a_{new} .

Qualification as a Candidate Element. We check the qualification of an element to be in the candidate set for both R_1 and R_2 . If an element a is dominated by a_{new} and the updated P_{new} value becomes below the probability threshold q , a is no longer a candidate element and will be deleted from R_1 or R_2 .

Qualification as a Probabilistic k -skyband Result. We check the qualification of an element as a probabilistic k -skyband result for elements in R_2 only. If an element a in R_2 survives the qualification test and $P_{skyband}(a)$ falls below q , it will be moved from R_2 to R_1 .

Insert a_{new} . $P_{skyband}$ value of a_{new} will be computed using the dynamic programming algorithm in Section 4. If $P_{skyband}(a_{new}) \leq q$, a_{new} is inserted into R_2 ; otherwise a_{new} is inserted into R_1 .

Algorithm 3. Insertion of (a_{new})

```

1 retrieve the element set  $D_{R_1}$  in  $R_1$  dominated by  $a_{new}$ ;
2 for each element  $e$  in  $D_{R_1}$  do
3   update  $P_{new}(e)$  using  $\lambda_{M_{new}}(e)$ ;
4   if  $P_{new}(e) < q$  then
5     delete  $e$  from  $R_1$ ;
6 retrieve the element set  $D_{R_2}$  in  $R_2$  dominated by  $a_{new}$ ;
7 for each element  $e$  in  $D_{R_2}$  do
8   update  $P_{new}(e)$  using  $\lambda_{M_{new}}(e)$ ;
9   if  $P_{new}(e) < q$  then
10    delete  $e$  from  $R_1$ ;
11 else
12   update  $P_{skyband}(e)$  using  $\lambda_{M_{skyband}}(e)$ ;
13   if  $P_{skyband}(e) < q$  then
14     move  $e$  from  $R_2$  to  $R_1$ ;

```

Algorithm 3 reports the steps to handle the insertion of a_{new} . Lines 1 to 5 handles the updates on R_1 where we need to consider the qualification as a candidate element only, while lines 6 to 14 tackles updates on R_2 where we need to handle both candidate and k -skyband result qualification. Note that retrieving the element set which are dominated by a_{new} can be efficiently implemented using existing techniques [9].

5.3 Handling Removal of a_{old}

With the expiration of a_{old} , we check the elements in R_1 and R_2 which are dominated by a_{old} and update their $P_{skyband}$ values since the $P_{skyband}$ values for these element will increase. For every element in R_1 and R_2 which is dominated by a_{old} , its $P_{skyband}$ value will be recomputed using the dynamic programming algorithm in Section 4. If the updated $P_{skyband}$ value for an element in R_1 becomes larger than or equal to q , it will be moved from R_1 to R_2 . Removing a_{old} is straightforward since we only need to delete it from R_1 or R_2 . Algorithm 4 depicts the key steps in handling the removal of a_{old} . Lines 2 to 6 handles the updates on R_1 and Lines 7 to 9 handles the updates on R_2 .

Algorithm 4. Removal of (a_{old})

- 1 delete a_{old} from R_1 or R_2 ;
 - 2 retrieve the elements set D_{R_1} in R_1 dominated by a_{old} ;
 - 3 **for each element** e in D_{R_1} **do**
 - 4 recompute $P_{skyband}(e)$;
 - 5 **if** $P_{skyband}(e) \geq q$ **then**
 - 6 move e from R_1 to R_2 ;
 - 7 retrieve the elements set D_{R_2} in R_2 dominated by a_{old} ;
 - 8 **for each element** e in D_{R_2} **do**
 - 9 recompute $P_{skyband}(e)$;
-

5.4 Continuously Processing Probabilistic k -Skyband Query

As introduced above, R-tree R_2 keeps the probabilistic k -skyband results for the current sliding window and is updated along the sliding of the window (i.e., new elements arrival and old elements expiration). So at every timestamp, scanning using breadth first search over R_2 and outputting all elements in R_2 provides the probabilistic k -skyband result for the current window.

6 Performance Evaluation

In this section, we only evaluate our techniques since this is the first paper studying the problem of probabilistic k -skyband computation over sliding windows.

All algorithms are implemented in C++ and compiled by GNU GCC. Experiments are conducted on PCs with Intel Xeon 2.4GHz dual CPU and 4G memory under Debian Linux. Our experiments are conducted on both real and synthetic datasets.

Real dataset is extracted from the stock statistics from NYSE (New York Stock Exchange). We choose 2 million stock transaction records of Dell Inc. from *Dec 1st 2000* to *May 22nd 2001*. For each transaction, the average price per volume and total volume are recorded. This 2-dimensional dataset is referred to as *stock* in the following. We randomly assign a probability value to each transaction; that is, probability values follows *uniform* distribution. Elements' arrival order is based on their transaction time. **Synthetic datasets** are generated as follows. We first use the methodologies in [2] to generate 2 million data elements with the dimensionality from 2 to 5 and the spatial location of data elements follow two kinds of distributions, *independent* and *anti-correlated*. Then, we use two models, *uniform* or *normal* distribution, to randomly assign occurrence probability of each element to make them be uncertain. In *uniform* distribution, the occurrences probability of each element takes a random value between 0 and 1, while in the *normal* distribution, the mean value P_μ varies from 0.1 to 0.9 and variance is set 10. We assign a random order for elements' arrival in a data stream.

Table 2 summarizes parameters and corresponding default values. **In our experiments, all parameters take default values unless otherwise specified.**

Table 2. System Parameters

Notation	Definition (Default Values)
n	Number of points in the dataset (2M)
N	Sliding Window size (400K)
d	Dimensionality of the of the dataset (3)
D	Dataset (Anti)
D_P	Probabilistic distribution of appearance (<i>uniform</i>)
P_μ	expected appearance probability (0.5)
q	probabilistic threshold (0.3)
k	k -skyband width (4)

6.1 Evaluate Space Efficiency

We evaluate the space usage in terms of the number of uncertain elements kept in $S_{N,q}$ against different settings. As this number may change as the window slides, we record the maximal value over the whole stream. Meanwhile, we also keep the maximal number of k -skyband.

The first set of experiments is reported in Figure 2 where 4 datasets are used: Inde-Uniform (Independent distribution for spatial locations and Uniform distribution for occurrence probability values), Anti-Uniform, Anti-Normal, and Stock-Uniform. We record the maximum sizes of $S_{N,q}$ and k -skyband. It is shown that very small portion of the 2-dimensional dataset needs to be kept. Although this proportion increases with the dimensionality rapidly, our algorithm can still achieve a 72% space saving even in the worst case, 5 dimensional *anti-correlated* data. Size of k -skyband is much smaller than that of candidates. Since the *anti-correlated* dataset is the most challenging, it will be employed as the default dataset in the following.

The second set of experiment evaluates the impact of sliding window size N on the space efficiency. As depicted in Figure 3(a), the space usage is sensitive towards the increment of window size.

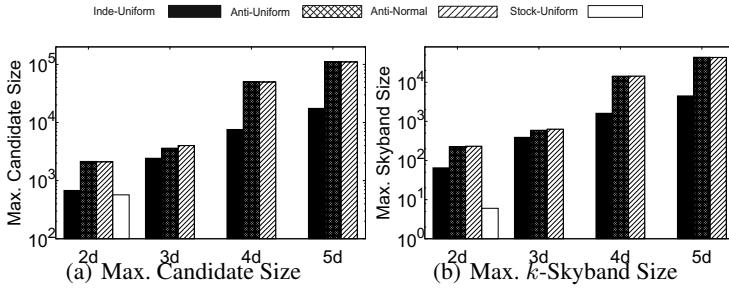
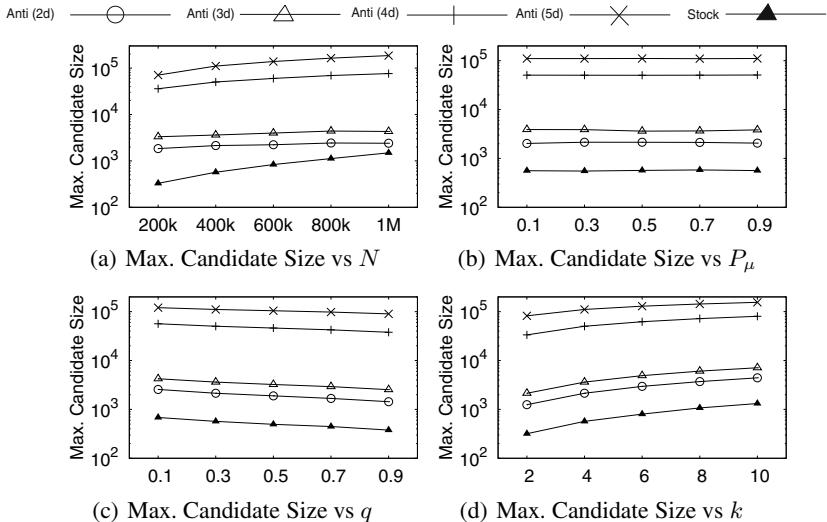
**Fig. 2.** Space Usage vs Diff. Data set**Fig. 3.** Space Usage

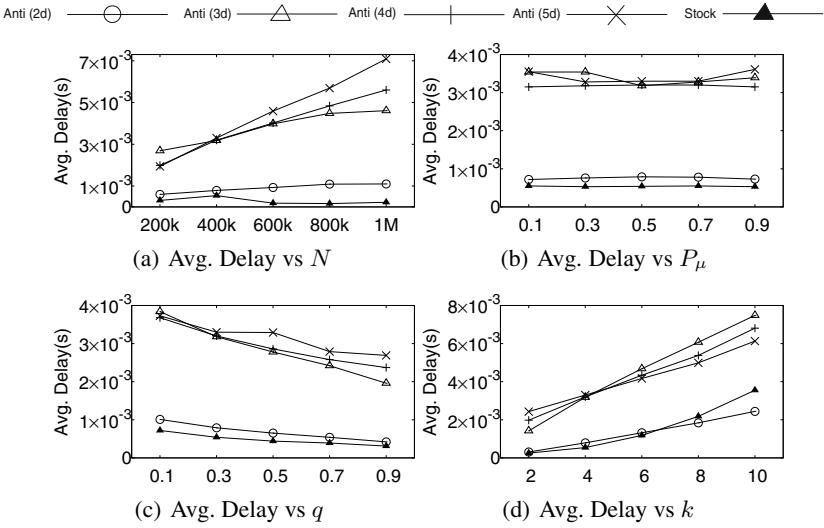
Figure 3(b) reports the impact of occurrence probability distribution against the space usage on different datasets. It demonstrates that there's no obvious relationship between candidate size and expected appearance probability.

Figure 3(c) reports the effect of probabilistic threshold q on space efficiency. The threshold probability q varies from 0.1 to 0.9. As expected, candidate set size drops as q increases.

Figure 3(d) reports the effect of k -skyband width k on space efficiency. As expected, candidate set size increases as k increases.

6.2 Evaluation Time Efficiency

We evaluate the time efficiency of our continuous query processing techniques. We first compare our algorithm with a trivial algorithm based on dynamic programming to

**Fig. 4.** Time Efficiency

compute $P_{skyband}$. We find it is too late since it recalculates $P_{skyband}$ value of every element in every sliding window. Thus, we exclude the trivial algorithm from further evaluation.

Since the processing time of one element is too short to capture precisely, we record the average time for each batch of 1K elements to estimate the delay per element.

Figure 4(a) evaluates the system scalability towards the size of the sliding window. The performance of our algorithm decreases as the size of sliding window increases. This is because the candidate size increases with N , as reported in Figure 3(a).

Figure 4(b) evaluates the impact of occurrence probability distribution on time efficiency of our algorithm where normal distribution is used for probability values. Similar to Figure 3(b), there is no obvious relationship.

Figure 4(c) evaluates the effect of probability threshold q . Since both size of candidate set and k -skyband objects set are small when q is large as depicted in Figure 3(c), our algorithm is more efficient when q increases.

The last experiment evaluates the impact of k -skyband width k on time efficiency. Results are reported in Figures 4(d). As expected, Figure 4(d) shows that processing cost increases when k increases.

6.3 Summary

As a short summary, our performance evaluation indicates that we only need to keep a small portion of stream objects in order to compute the probabilistic k -skyband over sliding windows. Moreover, our continuous query processing algorithms are very efficient and can support data streams with high speed for 2d and 3d datasets. Even for the most challenging data distribution, *anti-correlated*, we can still support the data stream with medium speed of more than 140 elements per second when dimensionality is 5.

7 Related Work

We review related work in two aspects, skyband and uncertain data streams. To the best of our knowledge, this paper is the first one to address the problem of k -skyband queries on uncertain data streams.

The concept of k -skyband is first proposed in [12]. [11] shows that it suffices to keep the k -Skyband of the data set to provide the top- k ranked queries for the monotone preference function f . Reverse k -skyband query is recently studied in [10].

Aggregates over uncertain data streams have been studied in [3,5,6]. Problems such as clustering uncertain data stream [1], frequent items retrieval in probabilistic data streams [15], and sliding window top- k queries on uncertain streams [7] are also investigated. The skyline query processing on uncertain data is firstly approached by Pei *et al* [13] where *Bounding-pruning-refining* techniques are developed for efficient computation. Lian *et al* [8] combine reverse skylines [4] with uncertain semantics and model the *probabilistic reverse skyline* query in both monochromatic and bichromatic fashion. Efficient pruning techniques are developed to reduce the search space for query processing. Probabilistic skyline processing over uncertain sliding windows are investigated in [16].

8 Conclusion

In this paper, we investigate the problem of probabilistic k -skyband query processing over sliding windows in a probability threshold fashion. We firstly characterize the minimum information to be kept for correct probabilistic k -skyband computation. Then a novel dynamic programming based algorithm is proposed to compute the k -skyband probabilities for a given uncertain element. Efficient, aggregate information enhanced, incremental updating techniques are developed to handle frequent insertions and deletions in the streaming environment. Time and space efficiency of our proposed techniques are demonstrated in a comprehensive empirical study over both real and synthetic datasets.

References

1. Aggarwal, C.C., Yu, P.S.: A framework for clustering uncertain data streams. In: ICDE 2008 (2008)
2. Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE 2001 (2001)
3. Cormode, G., Garofalakis, M.: Sketching probabilistic data streams. In: SIGMOD 2007 (2007)
4. Dellis, E., Seeger, B.: Efficient computation of reverse skyline queries. In: VLDB 2007 (2007)
5. Jayram, T., Kale, S., Vee, E.: Efficient aggregation algorithms for probabilistic data. In: SODA 2007 (2007)
6. Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating statistical aggregates on probabilistic data streams. In: PODS 2007 (2007)
7. Jin, C., Yi, K., Chen, L., Yu, J.X., Lin, X.: Sliding-window top- k queries on uncertain streams. In: VLDB 2008 (2008)

8. Lian, X., Chen, L.: Monochromatic and bichromatic reverse skyline search over uncertain databases. In: SIGMOD 2008 (2008)
9. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: Efficient skyline computation over sliding windows. In: ICDE 2005 (2005)
10. Liu, Q., Gao, Y., Chen, G., Li, Q., Jiang, T.: On efficient reverse k -skyband query processing. In: Lee, S.-G., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 544–559. Springer, Heidelberg (2012)
11. Mouratidis, K., Bakiras, S., Papadias, D.: Continuous monitoring of top- k queries over sliding windows. In: SIGMOD 2006 (2006)
12. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal progressive algorithm for skyline queries. In: SIGMOD 2003 (2003)
13. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: VLDB 2007 (2007)
14. Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. In: TKDE 2006 (2006)
15. Zhang, Q., Li, F., Yi, K.: Finding frequent items in probabilistic data. In: SIGMOD 2008 (2008)
16. Zhang, W., Lin, X., Zhang, Y., Wang, W., Yu, J.X.: Probabilistic skyline operator over sliding windows. In: Information Systems 2012 (2012)

SATURN: A Fast Keyword k NN Search System in Road Networks

Nan Zhang¹, Yang Wang², and Jianhua Feng¹

¹ Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, China

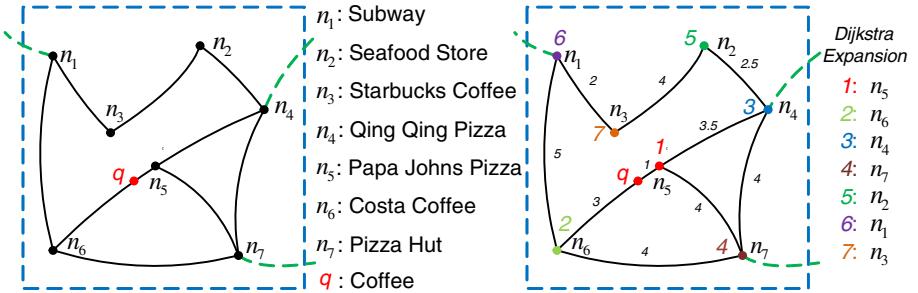
² National Computer Network Emergency Response Technical Team/
Coordination Center of China, 100029

n-zhang10@mails.tsinghua.edu.cn, aaron@ncic.ac.cn, fengjh@tsinghua.edu.cn

Abstract. Location-based services (LBS) have became more and more popular nowadays since people equipped with smart phones. Existing keyword k -nearest neighbor (k NN) search methods focus more on the keywords; therefore, they use direct distance of two points, also known as, Euclidean distance as spatial constraints. However, the nearest point-of-interest (POI) returned by these services may not be the nearest on the road networks. For some services that consider the road networks, they use road expansion methods to solve this problem. The speed limitation for a large road network and index structures for millions POI may be the bottlenecks for these services. To address those problems, we develop a fast keyword k NN search system in road networks, called SATURN. Instead of using road expansion methods, we introduce a grid-based shortest path computation method, a filter-and-verification framework to search fast in road networks, and we also devise an improvement of the grid index to further improve the performance. We conduct extensive experiments on real data sets, and the experimental results show that our method is efficient and scalable to large data sets, significantly outperforming state-of-the-art methods.

1 Introduction

As the fast development of smart phones, people nowadays have more chances to access their current locations with the help of GPS and WiFi localization system. Using their locations, users can preform the keyword k -nearest neighbor (k NN) search [1,2,3,4], that is, given a set of keywords, finding the top k nearest point-of-interests (POIs), considering both spatial and textual information. Existing works [1,5,6,7] focus mainly on Euclidean distance; therefore, they can easily use spatial data structures such as R-tree [8] to index the objects and perform search. However, it is quite inconvenient for users to use. For example, in Figure 1, there are seven POIs from n_1 to n_7 and a user entering a query “Coffee” at location q . n_3 and n_6 are the result candidates. If the user wants top one result, traditional services will return n_3 to the user since the Euclidean distance between q and n_3 is shorter. Nevertheless, in the road network, n_6 is much closer to the user; therefore, the service should give n_6 to the user.

**Fig. 1.** Road Networks

In this paper, we present a fast keyword k NN search system in road networks, called SATURN, in order to give users more convenient services. The main challenge for keyword k NN search in road networks is to achieve high interactive speed when the road networks are complicated. For a city with millions of POIs and complex road networks, users focus more on how to get the results fast and accurately. Existing algorithms [2] use Dijkstra-based [9] algorithms to expand the road networks from the query point. In this way, they can access all the POIs by the distance order and then considering the textual constraints in order to generate the final results. These methods will be greatly limited by the road network's topology. With a complicated city map, those methods are rather expensive as they involve a lot of useless computation on the expansion of road networks. Therefore, they cannot be adapted to the complicated city maps directly. To solve this problem, we introduce an effective grid-based shortest path computation framework to reduce the time cost on the computation of the shortest path. We also develop a novel filter-and-verification framework and effective prune technologies to facilitate the keyword k NN search. As an improvement, we devise a hybrid filtering algorithm based on grid index to further improve the performance. To summarize, we make the following contributions.

- We design and implement a fast keyword k -nearest neighbor search system in road networks, called SATURN, which has been commonly used and widely accepted.
- We develop a grid-based index structure and a shortest path computation framework. The shortest path framework effectively reduces the computation cost and improves the search experiences.
- We devise a novel filter-and-verification framework and effective prune technologies to facilitate the keyword k NN search. To utilize both spatial and textual information, we generate an improved hybrid filtering algorithm.
- We have conducted an extensive set of experiments and results show that SATURN is much more efficient than state-of-the-art methods and the indexing mechanism performs well.

The rest of this paper is organized as follows. Section 2 formulates the keyword k NN search problem and gives a baseline method. We introduce the grid-based

filter-and-verification framework in Section 3 and the hybrid filtering improvement in Section 4. Experimental results are provided in Section 5. We review the related work in Section 6, and make a conclusion in Section 7.

2 Preliminaries

2.1 Problem Formulation

Road Data. Formally, our paper considers a set of **node**, $\mathcal{N} = \{n_1, n_2, \dots, n_{|\mathcal{N}|}\}$, as the nodes that appear in road networks. Each $n \in \mathcal{N}$ consists a **location** $n.x, n.y$ where $n.x$ is the x-coordinate and $n.y$ is the y-coordinate of the node. Therefore, a node is denoted by $n = \langle n.x, n.y \rangle$. We also consider a set of **way**, $\mathcal{W} = \{w_1, w_2, \dots, w_{|\mathcal{W}|}\}$, as the ways in road networks. For each $w \in \mathcal{W}$, there are two nodes: the **start node** n_s and the **end node** n_e . There is a **weight** l , denoting the length of the way. Thus, a way is denoted by $w = \langle \langle n_s, n_e \rangle; l \rangle$. The nodes and ways are the basic elements of the road networks. We denote the $\text{dist}(n_i, n_j)$ as the shortest path of two nodes in road networks.

POI. Consider a set of **POIs**, $\mathcal{P} = \{p_1, p_2, \dots, p_{|\mathcal{P}|}\}$. Each $p \in \mathcal{P}$ has a **location** $\langle p.x, p.y \rangle$, indicating the x and y coordinate of the POI. p also consists a set of **keywords**, denoting by $p.k$. Therefore, a POI is denoted by $p = \langle \langle p.x, p.y \rangle; p.k \rangle$. Since all the POIs should be in road networks, we assign the nearest node's location as the POI's location. If there is no such node, we establish one and put the POI in that node. Therefore, a POI p is a special case of node n , where each p has an additional keyword set.

Query. A query q contains a **location** $\langle q.x, q.y \rangle$ as the user's location, a set of user-input **keywords** $q.k$ and a **number** k , which users can specify the top- k results returning. Thus, a query q is denoted by $q = \langle \langle q.x, q.y \rangle; q.k; k \rangle$.

Answer. We first denote the textual constraint by $\text{sim}(p_i.k, q.k)$, where sim is defined as the Jaccard similarity coefficient of the two keyword sets.

$$\text{sim}(p_i.k, q.k) = \frac{|p_i.k \cap q.k|}{|p_i.k \cup q.k|} \quad (1)$$

The answer for keyword k -nearest neighbor search finds all the $p_i \in \mathcal{P}$ such that $\text{sim}(p_i.k, q.k) > \tau$ and $\text{dist}(p_i, q)$ are the shortest k -th among all the POIs that satisfy the textual constraints.

Definition 1 (Keyword k -Nearest Neighbor Search in Road Networks). *Consider a road network \mathcal{R} , a set of POIs \mathcal{P} in that road network, and a query $q = \langle \langle q.x, q.y \rangle; q.k; k \rangle$. It returns the subset \mathcal{P}_q^k of \mathcal{P} with k POIs such that $\forall p \in \mathcal{P}_q^k$ and $\forall p' \in \mathcal{P} - \mathcal{P}_q^k$, $\text{dist}(p, q) \leq \text{dist}(p', q)$ and also $\text{sim}(p.k, q.k) > \tau$.*

For example, in Figure 1, given a $\tau = 0.5$, $k = 2$ and a query q showing in the figure, the results should be n_6 : “Costa Coffee” and n_3 : “Starbucks Coffee”.

2.2 Baseline Method

To solve any problem related to the shortest path problem, we can immediately come up with the Dijkstra methods, that is, expanding the road networks from the query location. Since users want to find the k -nearest POIs, the Dijkstra expansion can easily solve the problem. When a new node is expanding, for all the POIs at that location, the textual constraints are calculated. If the result is larger than the given τ , then we put the POI into a priority queue to be the result candidate. If the result is smaller than the given τ , the POI will not be the final result, we eliminate it. The algorithm will be terminated when the number of POIs in the queue equals to the k requirement. The correctness can be guaranteed because the node that is expanding must be the current shortest to the query location. Therefore, any node that will be expanded later must have a longer distance to the query location than the current node.

For example, in Figure 1, we set the $\tau = 0.5$. We first locate the position of q and begin expanding. Since the shortest node to q is n_5 , we test if n_5 consists certain POIs that satisfy the similarity constraints. The second node we expand is n_6 and it includes a keyword “Coffee”; the $sim = 0.5 \geq \tau$; therefore, it is a candidate. Then, the n_4, n_7, n_2, n_1 and n_3 are expanded in this order. Since n_3 has a $sim = 0.5 \geq \tau$, it is also a candidate.

However, the computation cost for a complicated map with millions nodes, ways and POIs is really expensive. For the worst case, we need to iterate the whole map and compute all the textual constraints and finally generate the results. For metropolitan-scale usage, we need to generate fast search algorithm and indexing structures to prune unnecessary nodes and POIs. To achieve high interactive speed, in this paper, we propose a grid-based search method.

3 SATURN Algorithm

In this section, we first introduce a filter-and-verification framework to solve the keyword k NN problem and then present our grid-based shortest path computation method, which is the key process in the verification phase of our algorithm.

3.1 Filter-and-Verification Framework

To answer the keyword k NN search, we want to prune unnecessary POIs that do not have similar textual information and then generate the final results. To this end, we introduce the filter-and-verification framework.

Filter. In the filter phase, we prune unnecessary POIs and nodes in road networks to generate a candidate set, which contains all the POIs that have textual similarity larger than the threshold τ . This is the textual filter phase.

Verification. We then verify the candidates generated by the previous phase by checking their shortest distance to the query location. We find the top- k POI results based on their location. This is the spatial verification phase.

3.2 Search Algorithm

Based on the filter-and-verification framework, we present the search algorithm.

Indexing Structure. We first build a keyword inverted index on top of the road networks to avoid iterating all the POIs. Formally, we denote the inverted index by \mathcal{I} and for a keyword w , its inverted index is \mathcal{I}_w . For example, in the road networks showing in Figure 1, the keyword “**Pizza**” appears in n_4 , n_5 and n_7 ; therefore, the three nodes are in the inverted index. By generating such an index, we can easily acquire all the nodes that have the keyword.

Generating Candidates. When a query comes, we first access all the keywords in the query and get the inverted index \mathcal{I}_w from our index. Since a query may contain several keywords, for example, “**Starbucks Coffee**”, we should merge all the inverted index \mathcal{I}_w to eliminate the repetitive POIs. After the merging step, we can get a set of POIs, which are the candidates that include the keywords in the query.

Filtering Candidates. Since we have acquired all the POIs that contain the keywords, we need one more step to filter all the POIs that do not satisfy the textual similarity constraint. For each POI, we calculate the textual similarity value using Equation 1. After the filter phase, we generate a set of POIs that satisfy the constraint and can be the final result.

Verifying Candidates. For each candidate, we need to verify whether it belongs to the top- k result set or not. To this end, we need to calculate the shortest distance of each candidate to the query location. We present a grid-based shortest path computation framework in Section 3.3, called GRASP (Grid-based Shortest Path). GRASP can calculate the road distance fast and accurately. After computing all the distances, we put the results in a min-heap and generate the top- k results to the user.

POI Pruning Technology. From the definition of road distance $R(*)$ and Euclidean distance $E(*)$, we see that road distance is not smaller than the Euclidean distance, $R(*) \geq E(*)$; therefore, we can use the Euclidean distance as a bound to prune unnecessary POIs. Since we want to find top- k results, we first sort all the POI candidates generated from filter phase based on their Euclidean distance. During the search, we maintain k POIs in the heap. For the current POI, denoted by p_c , that needs to calculate the distance, if the k -th POI, denoted by p_k , in the heap has a shorter road distance than the p_c ’s Euclidean distance, $R(p_k) < E(p_c) \leq R(p_c)$. Because $R(p_k) < R(p_c)$, p_c cannot be added to the heap. For the POIs after p_c , since they have a greater Euclidean distance, they cannot be added to the heap as well. Therefore, the search process can be terminated and the POIs in the heap are the final results that we need.

For example, if the $R(p_k) = 150$ and $E(p_c) = 130$, then $R(p_c)$ can be any value not smaller than 130, it has a good chance smaller than 150. So we need to calculate the distance to see if it can be added into the heap. While if $E(p_c) = 170$, then $R(p_c)$ must be greater than 150 and cannot be added into the heap.

Algorithm 1. SATURN Algorithm

Input: q : A query $q = \langle\langle q.x, q.y \rangle; q.k; k \rangle$; \mathcal{I} : A collection of inverted indexes
Output: \mathcal{R} : A result set of POIs ordering by road distance
begin

```

    Initiate a set, a heap and a list;  

    for  $w$  in  $q.k$  do  

         $\quad$  set.ADD( $\mathcal{I}.\text{get}(w)$ );  

    for  $p$  in set do  

         $\quad$   $sim = \text{CALTEXTUALSIM}(q.k, p.k)$ ;  

         $\quad$  if  $sim > \tau$  then  

             $\quad$   $\quad$  list.ADD( $p$ );  

    SORTPOIs(list);  

    for  $p$  in list do  

         $\quad$  if  $\text{EuclideanDistance}(p) < \text{heap.great}$  then  

             $\quad$   $dis = \text{GRASP}(p)$ ;  

             $\quad$  if  $dis < \text{heap.great}$  then  

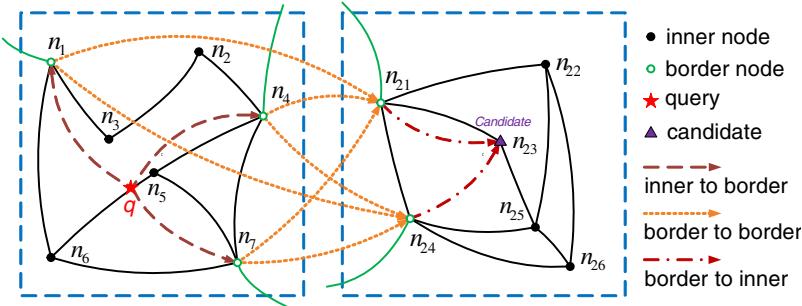
                 $\quad$   $\quad$  ENHEAP(heap,  $p$ );  

         $\quad$  else  

             $\quad$   $\quad$  break;  

    return heap;

```

Fig. 2. SATURN Algorithm**Fig. 3.** Grid-based Shortest Path

We give the pseudo-code of our SATURN algorithm in Figure 2. SATURN first initiates a set to maintain the POIs that have the same keywords, a list for candidates that passed the filter phase and a min-heap for the final results (line 1). It then merges the POIs from inverted index (line 1) and filters the candidates using the textual similarity equation (line 1). Then it sorts all the POI candidates with the Euclidean distance (line 1). Finally, in the verification phase, if the Euclidean distance is smaller than the k -th POI in the heap, we calculate the actual road distance (line 1) and verify if it can be the result or not (line 1).

The algorithm will be terminated when the Euclidean distance is larger as we mentioned before (line 1).

3.3 Grid-Based Shortest Path

In this section, we present the Grid-based Shortest Path computation framework.

Grid Indexing. For our GRASP algorithm, we use grid-based index structures. We first partition the road network into $m * n$ grids, denoted by $\mathcal{G}_{m,n}$. Each grid includes a set of nodes, ways between nodes and a set of POIs. In this way, we partition the map into $m * n$ small parts to fasten the computation of shortest path. For each $\mathcal{G}_{m,n}$, there are two kinds of nodes:

- **inner node**: its all adjacent nodes are in the same grid with it.
- **border node**: some adjacent nodes are in other grids.

After partition, we calculate some additional information for shortest path computation. We need to calculate distances of all the border node pairs. In each grid, we also compute the distance of inner node to the border node.

In Figure 3, we can see two grids. There are three border nodes n_1 , n_4 and n_7 in the left grid while two border nodes n_{21} and n_{24} in the right grid. We need to pre-calculate the distance between all these border nodes and in each grid, the distances of inner nodes to the border nodes, for example, n_5 to n_1 , n_4 , n_7 .

Shortest Path. There are two situations in our framework: at least one node is border node or two nodes are all inner nodes. For the first situation, it can be easily accessed if one node or two nodes are the border nodes since we have already indexed the distance.

For the second situation that two nodes are all inner nodes in one grid, we can use a Dijkstra expansion since the nodes number are limited. For example, if we partition the Beijing road network into $50 * 50$ grids, each grid will only have no more than a hundred nodes and we can ignore the time cost for such a small map.

The most complicated situation is two nodes are all inner nodes but in different grids. We use dynamic programming with a three-step-jump method to calculate. We can easily see that first we should jump from one inner node to its border and then jump from a border node to the other border node in the different grid. The third jump is from the border node to the other inner node. After each jump, we calculate and save the shortest path from q to the node. Therefore, at the last step, the final result would be the shortest path from q .

From the example showing in Figure 3, both the query and candidate are inner node. It first jumps from q to the border nodes and the distances are saved in each node n_1 , n_4 , n_7 separately. In the second step, it updates the distance to n_{21} and n_{24} . There are six ways from n_1 , n_4 , n_7 to n_{21} , n_{24} , that is, $n_1 \rightarrow n_{21}$, $n_4 \rightarrow n_{21}$, $n_7 \rightarrow n_{21}$, $n_1 \rightarrow n_{24}$, $n_4 \rightarrow n_{24}$, and $n_7 \rightarrow n_{24}$, saving the shortest distance in n_{21} , n_{24} . In the last step, it calculates the distance from n_{21} , n_{24} to candidate c and updates the final distance of c .

We give the pseudo-code of our GRASP algorithm in Figure 4. GRASP first initiates two lists to save the border nodes in two grids (line 2). It then locates

Algorithm 2. GRASP Algorithm

Input: q : A query location; c : A candidate location; \mathcal{D} : Pre-calculated distances

Output: dis : The shortest road distance between q and c
begin

```

    Initiate a  $list_1$ ,  $list_2$ ;
     $g_1 = \text{LOCATEINGRID}(q)$ ;  $g_2 = \text{LOCATEINGRID}(c)$ ;
     $list_1.\text{ADD}(g_1.\text{border})$ ;  $list_2.\text{ADD}(g_2.\text{border})$ ;
    for  $n_i$  in  $list_1$  do
         $n_i.\text{dis} = \mathcal{D}(q, n_i)$ ;
    for  $n_i$  in  $list_1$  do
        for  $n_j$  in  $list_2$  do
            if  $n_i.\text{dis} + \mathcal{D}(n_i, n_j) < n_j.\text{dis}$  then
                 $n_j.\text{dis} = n_i.\text{dis} + \mathcal{D}(n_i, n_j)$ ;
    for  $n_j$  in  $list_2$  do
        if  $n_j.\text{dis} + \mathcal{D}(n_j, c) < c.\text{dis}$  then
             $c.\text{dis} = n_j.\text{dis} + \mathcal{D}(n_j, c)$ ;
    return  $c.\text{dis}$ ;

```

Fig. 4. GRASP Algorithm

the query and candidate in the grids (line 2), putting all the border nodes in each grid (line 2). In the first step (line 2), it updates border distances in grid one from query. In the second step (line 2), it updates border distances of two grids. In the third step (line 2), it updates the distance from border to candidate. The shortest distance from query to candidate will be returned finally (line 2).

4 SATURN Algorithm Improvement

In this section, we present an improvement of the SATURN algorithm by taking consideration of both textual and spatial information simultaneously in the filter phase and propose a grid prune technology to further improve the performance.

4.1 Hybrid Filtering Algorithm

In the filter step of SATURN algorithm, we only consider the textual information to generate the candidate POIs. However, to further improve the performance, we can simultaneously utilize textual and spatial information to prune unnecessary regions and POIs. Since we want to find top- k nearest results, we can filter the candidates from the nearest grid to the further ones.

Grid Distance. We first give the definition of grid distance. For two grids, g_i and g_j , for all the road distance from node in g_i to node in g_j , the shortest one is the grid distance $g_{ij}.\text{dis}$. We add the grid distance information to our indexing structure in order to prune unnecessary grids during the filter phase.

Algorithm 3. Improved SATURN Algorithm

Input: q : A query $q = \langle\langle q.x, q.y \rangle; q.k; k \rangle$; \mathcal{G} : Grid indexes;
Output: \mathcal{R} : A result set of POIs ordering by road distance
begin

```

    Initiate a list, a heap;
     $g_q = \text{LOCATEINGRID}(q)$ ;
    list =  $\text{SORTGRID}(g_q, \mathcal{G})$ ;
    for  $g_i$  in list do
        if  $\text{FILTERGRID}(g_i.K)$  then
            for  $p_j$  in  $g_i$  do
                 $sim = \text{CALTEXTUALSIM}(q.k, p_j.k)$ ;
                if  $sim > \tau$  and  $\text{EuclideanDistance}(p_j) < \text{heap.great}$  then
                     $dis = \text{GRASP}(p_j)$ ;
                    if  $dis < \text{heap.great}$  then
                         $\text{ENHEAP}(\text{heap}, p_j)$ ;
                else
                    break;
    return heap;

```

Fig. 5. Improved SATURN Algorithm

Sorting Grids. The algorithm first locates the query in the grid g_q . It then sorts all the grids in the map based on their grid distance to g_q . We filter the grids by this list in order to utilize the spatial information. The grids near the query can be filtered first; therefore, we can eliminate extra computation of POIs.

Filtering Keywords. Addition to the previous information, for each grid, we need to maintain an inverted index of keywords that appear in this grid. Therefore, when a query comes, we can filter a grid by textual information. If the grid does not contain the keywords in the query, we can immediately eliminate the grid and filter the other grids. If the grid consists the keywords, we access the POIs in the grid and use the textual similarity function to further filter the POIs, generating results for the next phase.

Verifying Candidates. For all the candidates in a certain grid, we verify if the results are the top- k POIs. We use GRASP to calculate the real road distance of each pair. Here, the SATURN POI pruning technology is still working that we use the Euclidean distance as a boundary to prune unnecessary POIs. After verifying all the candidates in a certain grid, the algorithm moves to the next grid and continue the filter-and-verification process.

Grid Pruning Technologies. Similar as the POI pruning technology in SATURN, we also have a grid pruning technology to terminate the algorithm. For the next grid g_x that need to be filtered, if the grid distance from g_x to g_q is greater than the k -th POI's distance to q in the heap, we can stop the algorithm

because any distance in g_x to g_q will be greater than k -th POI's distance. There is no need to compare the following grids at all.

We give the pseudo-code of our the improved SATURN algorithm in Figure 5. It first initiates a list to save the ordering grids and a heap for final results (line 3). It then locates the query in the grids (line 3) and sorts the grids by the grid distance to the query grid (line 3). For each grid, it uses grid filter to determine whether the algorithm can be terminated or not (line 3). For each POI in the grid, it utilizes both spatial and textual information to filter (line 3). Finally, it puts the POIs that passed the verification step into the heap and returns the results to the user (line 3).

5 Experiment

We have implemented the SATURN system and conducted a set of experiments on real data sets. We used the data of real cities. Table 1 shows the detailed information of the data sets.

SATURN Performance Evaluation. We compared our SATURN algorithm with the commonly used Dijkstra expansion algorithm [2] in three data sets to evaluate the search efficiency. We randomly generate 10,000 queries from the POI database, using the keywords as input query keywords, the location of POI as the query location. Figure 6 shows the results.

We see that the results of SATURN are approximately 100 times better than expansion algorithms. The reason is that during the search process, SATURN saves quite a lot of time on the calculation of the shortest path. The Dijkstra methods spend more time on access useless nodes and POIs when they expand the map. For SATURN, it needs only to access those candidate POIs that have passed the textual similarity constraints. The improved algorithm is about 10 times better than SATURN. This is because during the search, it can first prune useless grids with quite a large number of POIs.

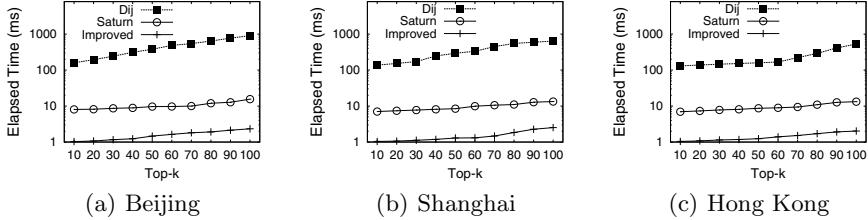
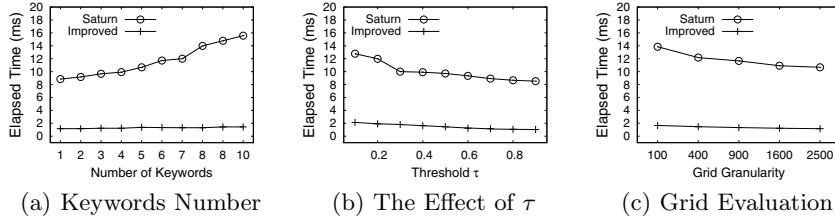
For the increasing value of k , the search time increases with it. The Dijkstra methods increase greatly because the search depends on the expansion and the more POIs it returns, the larger the map it expands. For our SATURN and improved method, the time increases little since we depend less on the topology of the road networks.

Parameter Evaluation. We first evaluate the effects of keywords number to SATURN and the improved algorithm. We set the number of keywords from one to ten, returning top-20 results and we set $\tau = 0.5$. Figure 7(a) shows the results. We see that SATURN increases with the keywords because the number of candidates increases and we need to calculate more textual similarity before we find the top- k . However, for the improved algorithm, it combines the spatial information at the filter step; therefore, the number of keywords affects it less.

We then evaluate the effects of the value τ , which is the threshold in the calculation of the textual similarity. We set the value of τ from 0.1 to 0.9 and $k = 20$. The results are shown in Figure 7(b). We can see that when τ increases,

Table 1. Data Sets

City Name	Nodes	Ways	POIs	Keywords
Beijing	135484	150718	1212353	6002327
Shanghai	127615	141803	948915	4930751
Hong Kong	129799	138389	839440	4165434

**Fig. 6.** Performance Comparison**Fig. 7.** Parameter Evaluation & Grid Granularity Evaluation

the number of textual candidates decreases; therefore, SATURN will perform better. However, the number of results will also decrease since the constraint is much tighter. It is possible that the number of POIs returning will be less than 20. Therefore, we should carefully choose the value of τ in the real system to balance the trade-off of time cost and results. For the improved algorithm, the tighter constraint also makes it calculate less POIs so the time declines slightly.

Grid Granularity Evaluation. We evaluate the effects of different grid granularity. We set the number of grids from 100 to 2500 in a city thus the length of grid is from 10 kilometers to 2 kilometers. Figure 7(c) shows the results. We see that the more grids we partition the road networks, the less search time will be. For SATURN, the time for calculating the distance will decrease in a single grid since the number of nodes are less. For the improved algorithm, it can prune more useless grids and find the candidates fast. However, for a road network with more grids, the index size will also increase because there will be more border nodes. For Beijing, 2500 grids index will use 940M additional space.

6 Related Work

Most existing works [1,5,6,7] consider Euclidean distance; therefore, they use R-tree [8] to index the spatial objects and use branch-and-bound method to search for the results. Based on the R-tree indexes, k -nearest neighbor [10,11,12,13] queries have been well studied in traditional databases. They use a heap to maintain the objects and expand the object with a mini-distance. However, in the real world usage, we need to consider the road network topology to fulfill the users' requirement. [14] formalizes the problem of k NN search in road networks and presents a system prototype for such queries. [2] uses algorithm based on Dijkstra's algorithm to perform k NN search. Thus when expanding the road, the computation cost is expensive. Papadias et al. [15,16,17] describe a framework that integrates network and Euclidean information in order to answer different spatial queries. The main disadvantage of this approach is a dramatic degradation in performance when the nodes that need to retrieve are great. [18] systematically studies the problem of network distance computation, which is the most fundamental problem in road network. Shahabi et al. [3] propose an embedding technique to transform a road network into a higher dimensional space. The main disadvantage is that it provides only an approximation of the actual distance. In [19,20], they use pre-computed nearest neighbors as index; however, they cannot incrementally retrieve the results. [4] uses the Voronoi-based approach, which tries to partition a large network into smaller Voronoi regions. The method will get computationally more expensive for increasing values of k .

7 Conclusion

In this paper, we present and implement a fast keyword k -nearest neighbor search system in road networks, called SATURN. In order to reduce the computation cost of the shortest path, we devise an effective grid-based shortest path (GRASP) computation framework. We also develop a filter-and-verification framework and effective POI prune technologies to facilitate the keyword k NN search. To further improve the performance, we devise a hybrid filtering algorithm based on grid indexes to simultaneously use the textual and spatial information to prune unnecessary grids. The results from experiments show that our system has the capability of dealing with massive data and has a better performance than the existing state-of-the-art systems.

References

1. Cheung, K.L., Fu, A.W.C.: Enhanced nearest neighbour search on the r-tree. SIGMOD Record 27(3), 16–21 (1998)
2. de Almeida, V.T., Güting, R.H.: Using Dijkstra's algorithm to incrementally find the k -nearest neighbors in spatial network databases. In: SAC, pp. 58–62 (2006)
3. Shahabi, C., Kolahdouzan, M.R., Sharifzadeh, M.: A road network embedding technique for k -nearest neighbor search in moving object databases. In: ACM-GIS, pp. 94–10 (2002)

4. Kolahdouzan, M.R., Shahabi, C.: Voronoi-based k nearest neighbor search for spatial network databases. In: VLDB, pp. 840–851 (2004)
5. Ferhatosmanoglu, H., Stanoi, I., Agrawal, D.P., El Abbadi, A.: Constrained nearest neighbor queries. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 257–276. Springer, Heidelberg (2001)
6. Berchtold, S., Ertl, B., Keim, D.A., Kriegel, H.P., Seidl, T.: Fast nearest neighbor search in high-dimensional space. In: ICDE, pp. 209–218 (1998)
7. Tao, Y., Papadias, D., Shen, Q.: Continuous nearest neighbor search. In: VLDB, pp. 287–298 (2002)
8. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD Conference, pp. 47–57 (1984)
9. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
10. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. ACM Trans. Database Syst. 24(2), 265–318 (1999)
11. Katayama, N., Satoh, S.: The sr-tree: An index structure for high-dimensional nearest neighbor queries. In: SIGMOD Conference, pp. 369–380 (1997)
12. Papadopoulos, A., Manolopoulos, Y.: Performance of nearest neighbor queries in r-trees. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 394–408. Springer, Heidelberg (1996)
13. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. In: SIGMOD Conference, pp. 71–79 (1995)
14. Jensen, C.S., Kolárvr, J., Pedersen, T.B., Timko, I.: Nearest neighbor queries in road networks. In: GIS, pp. 1–8 (2003)
15. Papadias, D., Zhang, J., Mamoulis, N., Tao, Y.: Query processing in spatial network databases. In: VLDB, pp. 802–813 (2003)
16. Papadias, D., Tao, Y., Mouratidis, K., Hui, C.K.: Aggregate nearest neighbor queries in spatial databases. ACM Trans. Database Syst. 30(2), 529–576 (2005)
17. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. IEEE Trans. Knowl. Data Eng. 17(6), 820–833 (2005)
18. Hu, H., Lee, D.L., Lee, V.C.S.: Distance indexing on road networks. In: VLDB, pp. 894–905 (2006)
19. Huang, X., Jensen, C.S., Šaltenis, S.: The islands approach to nearest neighbor querying in spatial networks. In: Medeiros, C.B., Egenhofer, M., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 73–90. Springer, Heidelberg (2005)
20. Cho, H.J., Chung, C.W.: An efficient and scalable approach to cnn queries in a road network. In: VLDB, pp. 865–876 (2005)

Top- K Aggregate Queries on Continuous Probabilistic Datasets

Jianwen Chen¹, Ling Feng¹, and Jun Zhang²

¹ Dept. of Computer Science & Technology, Tsinghua University, Beijing, China

chen-jw08@mails.tsinghua.edu.cn, fengling@tsinghua.edu.cn

² No. 145, Erqi Road, Jiangan District, Wuhan City, Hubei Prov.

junzhangx@126.com

Abstract. Top- K aggregate query, which ranks groups of tuples by their aggregate values and returns the K groups with the highest aggregates, is a crucial requirement in many domains such as information extraction, data integration, and sensor data processing. In this paper, we formulate the top- K aggregate queries when the tuple scores are presented as continuous probability distributions. Algorithms for top- K aggregate queries are presented. To further improve the performance, we develop pruning techniques and adaptive strategy that avoid computing the exact aggregate values of some groups that are guaranteed not to be in top- K . Our experimental study shows the efficiency of our techniques over several datasets with continuous attribute uncertainty.

1 Introduction

Due to the large amount of uncertain data emerging from a variety of application domains, probabilistic databases have received a lot of attentions recently [3,1]. For example, the rent of an automatically extracted apartment from the web may be missing and is assigned a range of possible prices [*low*, *upper*] with the uniform distribution. In a sensor network, the measurement may be inaccurate due to noises and is usually presented as a continuous probability distribution on possible values. Typically, to handle such uncertain data, an uncertain table is used to store a set of tuples each of which comes with a score value represented as a continuous probability density function (pdf).

On the continuous uncertain data, ranking groups of records by their aggregate values and returning the K groups with the highest aggregates arises naturally in many application domains. In the context of web, it may be required to group the extracted objects into categories and find the top- K categories based on aggregates on continuous uncertain attributes. In a sensor network, it may be required to group sensor readings by location to find the top- K locations based on average temperature where each temperature is presented as a continuous probability distribution. All these applications motivate the need for formal formulation and efficient processing techniques for top- K aggregate queries on continuous uncertain data, which is the aim of this paper.

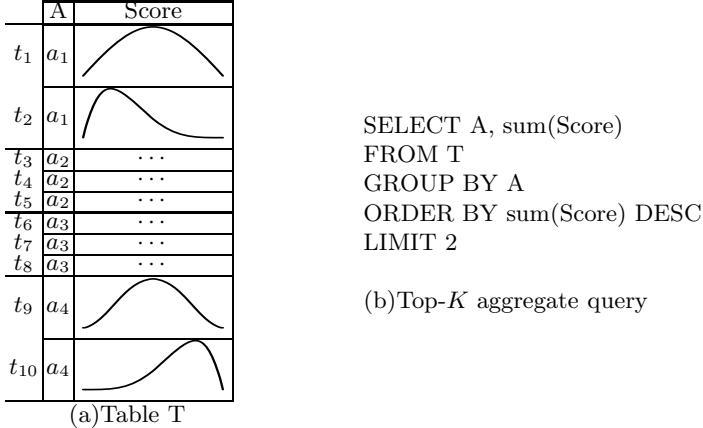


Fig. 1. An example top- K aggregate query issued on a table with score attributes described as continuous pdfs

Figure 1 illustrates a top- K aggregate query issued on a table with 10 tuples t_1, t_2, \dots, t_{10} each of which has a score attribute represented as a continuous probability density function (pdf). The semantic of the top- K aggregate query is first dividing all the tuples into four groups according to attribute A , resulting in $g_1 = \{t_1, t_2\}, g_2 = \{t_3, t_4, t_5\}, g_3 = \{t_6, t_7, t_8\}, g_4 = \{t_9, t_{10}\}$, and then returning the 2 groups with the largest **sum** aggregates. Since the scores of the tuples in each group are continuous random variables described as pdfs, the **sum** aggregate on them also produces a continuous random variable described as a pdf (called group aggregate pdf or aggregate pdf in the following). Suppose that the **sum** aggregate on groups g_1, g_2, g_3 , and g_4 leads to group aggregate pdfs ρ_1, ρ_2, ρ_3 , and ρ_4 respectively. The groups are ranked according to ρ_1, ρ_2, ρ_3 , and ρ_4 and the top-2 largest groups are returned to the user.

In this paper, the groups are ranked according to the parameterized ranking function (*PRF*) proposed in [7], which can simulate or approximate a variety of ranking functions with approximate parameter choices. In [7], a *generating functions* framework is used to compute the *PRF* values. This produces a cubic algorithm. Their work aims to rank tuples according to their scores, where the pdfs representing the scores are given directly. However, in our work, the pdfs which are used to compute the *PRF* values are derived from the aggregations of the tuples in each group, and thus the computing of the top- K aggregate query requires more time cost. In this paper, we present another equivalent method for computing the *PRF* values, which produces a quadratic algorithm. To further improve the performance, we develop pruning techniques and adaptive strategy that avoid computing the exact *PRF* values of some groups that are guaranteed not to be in top- K . Our experimental study shows the efficiency of our techniques over several datasets with continuous attribute uncertainty.

This paper is organized as follows. Section 2 reviews the related work on ranking uncertain data. Section 3 formally formulates the top- K aggregate queries

on continuous uncertain data. Section 4 presents the algorithms, including the basic algorithm and optimizations. Section 5 presents experimental results, and Section 6 concludes the paper.

2 Related Work

Recently, ranking uncertain data has drawn many research interests, with different ranking semantics proposed, such as tuple probability based rank [10], **U-Top** and **U-Rank** [13], probabilistic threshold top- K [5], **expected rank** [6], and **typical top- K** queries [4]. There are also other research work aiming to improve the ranking performance [16,14].

However, the aforementioned research work mainly focused on discrete uncertainty. Soliman and Ilyas [12] considered the problem of ranking continuous uncertain data, considering **UTop-Rank**, **UTop-Prefix**, **UTop-Set**, and **Rank-Agg** semantics. Probabilistic ranked (**PRank**) query and probabilistic inverse ranking (**PIR**) query are formulated and tackled in [15] and [8] respectively. Jian Li and Amol Deshpande [7] further presented a **PRF** based framework for ranking continuous uncertain data, which can simulate or approximate a variety of ranking functions with appropriate parameter choices. Cheng et al. [2] studied aggregate queries on continuous uncertain data, . However, top- K aggregate queires were not considered in their work.

Soliman and Ilyas [11] considered the top- K aggregate queries on discrete uncertain data. In this paper we focus on the top- K aggregate queries on continuous uncertain data. Note that the number of possible worlds corresponding to the discrete uncertainty is finite, while the number of possible worlds corresponding to the continuous uncertainty is uncountable. This makes the semantic formulation and efficient ranking techniques such as approximation and pruning for the top- K aggregate queries on discrete uncertain data cannot apply to the continuous uncertain data directly.

3 Problem Formulation

In this section, we formally formulate the top- K aggregate query when each tuple is associated with a score described as a continuous probability distribution.

Let T be a probabilistic dataset containing tuples t_1, t_2, \dots, t_n . Each tuple t_i has a corresponding score s_i described as a pdf (probability density function) f_i . All the scores are considered as independent continuous random variables. Given a set of grouping attributes A and a group aggregate function Agg (**sum**, **count**, **avg**, **max**, **min**), a **top- K aggregate query** first divides T into m groups $G = \{g_1, g_2, \dots, g_m\}$ according to grouping attributes A , and then computes the aggregate values for each group, and returns the K groups with the largest aggregate values.

For each group g containing $|g|$ tuples $t_1, t_2, \dots, t_{|g|}$ with scores $s_1, s_2, \dots, s_{|g|}$ described as probability density functions $f_1, f_2, \dots, f_{|g|}$, the aggregate values

for **sum**, **avg**, **max**, **min** are derived as probability density functions (called group aggregate pdf or aggregate pdf) from $f_1, f_2, \dots, f_{|g|}$, described as follows:

sum: The pdf of the **sum** of s_1 and s_2 can be computed as:

$$f_{s_1+s_2}(x) = \int_{-\infty}^{+\infty} f_1(y)f_2(x-y)dy \quad (1)$$

The pdf of the **sum** of $s_1, s_2, \dots, s_{|g|}$, denoted as $f_{s_1+s_2+\dots+s_{|g|}}$ can be computed inductively.

avg: The pdf of the **avg** of $s_1, s_2, \dots, s_{|g|}$ can be computed from $f_{s_1+s_2+\dots+s_{|g|}}$ as follows:

$$\frac{f_{s_1+s_2+\dots+s_{|g|}}}{|g|}(x) = |g|f_{s_1+s_2+\dots+s_{|g|}}(|g|x) \quad (2)$$

max: The pdf of the **max** of $s_1, s_2, \dots, s_{|g|}$ can be computed as:

$$f_{\max(s_1, s_2, \dots, s_{|g|})} = \sum_{1 \leq i \leq |g|} (f_i(x) \prod_{1 \leq j \leq |g|, j \neq i} \int_{-\infty}^x f_j(t)dt) \quad (3)$$

min: The pdf of the **min** of $s_1, s_2, \dots, s_{|g|}$ can be computed as:

$$f_{\min(s_1, s_2, \dots, s_{|g|})} = \sum_{1 \leq i \leq |g|} (f_i(x) \prod_{1 \leq j \leq |g|, j \neq i} (1 - \int_{-\infty}^x f_j(t)dt)) \quad (4)$$

The **count** aggregate has the same semantic as in the traditional database and simply returns the number of tuples in group g . Hence, in the following we will only focus on the above four aggregates.

After computing the aggregate for each group, the next step is to find the K groups with the largest aggregates and return to the user. Since the aggregate of each group is corresponding to a random variable described as a probability density function, there are a variety of different approaches to ranking the groups. In this paper, we adopt the parameterized ranking function (*PRF*) proposed in [7] to rank the groups.

The parameterized ranking function (*PRF*), $\gamma : G \rightarrow R$ is defined to be:

$$\gamma(g) = \omega_1 P(r(g) = 1) + \omega_2 P(r(g) = 2) + \dots + \omega_m P(r(g) = m) \quad (5)$$

where g is a group in $G = \{g_1, g_2, \dots, g_m\}$, $r(g)$ denotes the rank of g , $P(r(g) = i)$ is the probability that g is ranked at position i . $\omega_1, \omega_2, \dots, \omega_m$ are monotonically non-increasing real numbers that weight the probabilities that g is ranked at the corresponding positions. In the following, $\gamma(g)$ is called the *PRF* value of group g .

A top- K aggregate query returns the K groups with the highest *PRF* values. *PRFs* can approximate a variety of different ranking functions through appropriate choices of the weight parameter values. For example, when $\omega_1 = 1, \omega_2 = 0, \dots, \omega_m = 0$, all the groups are ranked by the probabilities that they have the largest aggregate; when $\omega_1 = 1, \omega_2 = 1, \dots, \omega_h = 1, \omega_{h+1} = 0, \dots, \omega_m = 0$, all the groups are ranked by the probabilities that they are ranked in the first h

places; when $\omega_1 = m - 1, \omega_2 = m - 2, \dots, \omega_m = 0$, the *PRF* values will produce the same ranking as the *expected ranks* [6].

In [7], a *generating functions* framework is proposed to compute the *PRF* value for ranking uncertain tuples. This produces a cubic algorithm. The real line is partitioned into small intervals such that the pdf of each tuple can be approximated as a single polynomial in each small interval. For each small interval I_j , let M_j be the set of tuples whose pdf support (the support of pdf f is defined as $\text{supp}(f) = \{x|f(x) > 0\}$) contains I_j and $m_j = |M_j|$, the time cost of computing the *PRF* value for each tuple is $O(\sum_j m_j^3)$.

Next, we shall describe another equivalent computing process which produces a quadratic algorithm and further inspires our pruning approach.

For groups $G = \{g_1, g_2, \dots, g_m\}$, we use $\gamma(g_1)$ as an example to illustrate the process of computing the *PRF* values. In the following, we use $\rho_1, \rho_2, \dots, \rho_m$ to denote the pdfs of the aggregates of g_1, g_2, \dots, g_m respectively.

$\gamma(g_1)$ can be computed as

$$\begin{aligned} & \gamma(g_1) \\ &= \omega_1 P(r(g_1) = 1) + \omega_2 P(r(g_1) = 2) + \dots + \omega_m P(r(g_1) = m) \\ &= \omega_1 \int_{-\infty}^{+\infty} \mathcal{F}_1^{(m)}(x) dx + \omega_2 \int_{-\infty}^{+\infty} \mathcal{F}_2^{(m)}(x) dx + \dots + \omega_m \int_{-\infty}^{+\infty} \mathcal{F}_m^{(m)}(x) dx \end{aligned} \quad (6)$$

where $\mathcal{F}_i^{(m)}(x)$ ($1 \leq i \leq m$) can be computed recursively as follows:

$$\mathcal{F}^{(1)}(x) = \rho_1(x) \quad (7)$$

and

$$\begin{aligned} \mathcal{F}_1^{(u+1)}(x) &= \mathcal{F}_1^{(u)}(x) \int_{-\infty}^x \rho_{u+1}(t) dt \\ \mathcal{F}_2^{(u+1)}(x) &= \mathcal{F}_1^{(u)}(x)(1 - \int_{-\infty}^x \rho_{u+1}(t) dt) + \mathcal{F}_2^{(u)}(x) \int_{-\infty}^x \rho_{u+1}(t) dt \\ &\dots \\ \mathcal{F}_u^{(u+1)}(x) &= \mathcal{F}_{u-1}^{(u)}(x)(1 - \int_{-\infty}^x \rho_{u+1}(t) dt) + \mathcal{F}_u^{(u)}(x) \int_{-\infty}^x \rho_{u+1}(t) dt \\ \mathcal{F}_{u+1}^{(u+1)}(x) &= \mathcal{F}_u^{(u)}(x)(1 - \int_{-\infty}^x \rho_{u+1}(t) dt) \end{aligned} \quad (8)$$

For any other group g_i in G , $\gamma(g_i)$ can be computed similarly by considering g_i as the first group, and the other groups are sorted arbitrarily. The order of the other groups does not affect the computing result for $\gamma(g_i)$. However, taking different orders may require different time costs. We shall describe the order of the groups adopted in our pruning based optimization in the next section. In the next section, we shall show that the time cost of computing the *PRF* value for each group g_i is $O(N_{\rho_i} m^2)$ where N_{ρ_i} denotes the number of subintervals of the support of ρ_i divided by δ in the Simpson method. We also show that the

computing process can be further simplified and the time costs are reduced to $O(N\rho_i|G_2|^2)$ where G_2 is the number of groups whose aggregate pdf support overlap with the aggregate pdf support of g_i .

In this paper, we assume that all the pdfs have bounded support. Hence, all the integrals in the above formulas defined with infinite integral limits can be transformed into equivalent integrals with finite limits. For example, suppose pdf f has a bounded support $[a, b]$, then $\int_{-\infty}^{+\infty} f(x)dx = \int_a^b f(x)dx$. If a random variable has a pdf with unbounded support, we truncate the distribution and ignore the tail with minuscule probability. Suppose X is a random variable with mean μ and variance σ^2 , we have $P(|X - \mu| \geq t\sigma) \leq \frac{1}{t^2}$ which can be derived from the Chebychev's Inequality. Hence, by selecting appropriate t , a bounded dominating range of the support can be decided.

4 Algorithms

The general framework for implementing a top- K aggregate query is first dividing all the tuples into groups, computing the aggregate for each group, and then returning the groups with the K largest PRF values. The process of dividing all the tuples into groups is the same as in the traditional database. In the following, we shall focus on computing the aggregates and selecting the K groups with the largest PRF values.

4.1 Basic Algorithm

Suppose that all the tuples in $T = \{t_1, t_2, \dots, t_n\}$ have been divided into groups $G = \{g_1, g_2, \dots, g_m\}$. The basic algorithm first computes the aggregates for each group, i.e., $\rho_1, \rho_2, \dots, \rho_m$, according to Equation (1), (2), (3), or (4), computes $\gamma(g_1), \gamma(g_2), \dots, \gamma(g_m)$ according to Equation (6), then selects the K largest, and returns the corresponding groups.

All the integrals are computed by the Simpson method. For a function $f : \mathcal{R} \rightarrow \mathcal{R}$ on an interval $[a, b]$, the integral of f can be computed as follows according to the Simpson method:

$$\int_a^b f(x)dx = \sum_{i=1}^n S_i(f) \quad (9)$$

where $[a, b]$ is divided into n subintervals with equal length δ , and in each subinterval $[x_i, x_{i+1}]$,

$$S_i(f) = \frac{x_{i+1} - x_i}{6} (f(x_i) + 4f(\frac{x_i + x_{i+1}}{2}) + f(x_{i+1})) \quad (10)$$

Note that all the integral limits $-\infty$ and $+\infty$ are transformed to the left end point and right end point of the support of the integrand. According to the definitions of the aggregates in Section 3, the supports of ρ_i for the **sum**, **avg**, **max**,

Table 1. Support of the aggregate pdf for group g_i

aggregate	support
sum	$[\sum_{j=1}^{ g_i } L_{supp(f_j)}, \sum_{j=1}^{ g_i } R_{supp(f_j)}]$
avg	$[\frac{\sum_{j=1}^{ g_i } L_{supp(f_j)}}{ g_i }, \frac{\sum_{j=1}^{ g_i } R_{supp(f_j)}}{ g_i }]$
max	$[\max_{j=1}^{ g_i } L_{supp(f_j)}, \max_{j=1}^{ g_i } R_{supp(f_j)}]$
min	$[\min_{j=1}^{ g_i } L_{supp(f_j)}, \min_{j=1}^{ g_i } R_{supp(f_j)}]$

and **min** aggregates are shown in Table 1. Here, we suppose that g_i contains tuples t_j ($1 \leq j \leq |g_i|$) with corresponding scores described as pdf f_j ($1 \leq j \leq |g_i|$). We use $supp(f_j)$ to denote the support of f_j and $L_{supp(f_j)}$ the left end point of $supp(f_j)$ and $R_{supp(f_j)}$ the right end point of $supp(f_j)$.

In the following, we use N_{f_i} and N_{ρ_i} to denote the number of subintervals of the supports of f_i and ρ_i divided by δ in the Simpson method. For the **sum** and **avg** aggregates, the time cost for computing ρ_i is $O(|g_i|^2 N_{avg}^2)$, where $|g_i|$ is the number of tuples in group g_i and N_{avg} is the average number of subintervals of the supports of f_i s divided by δ in the Simpson method. For the **max** and **min** aggregates, the time cost for computing ρ_i is $O(|g_i| N_{max})$, where $N_{max} = \max\{N_{\rho_i}, N_{f_1}, N_{f_2}, \dots, N_{f_{|g_i|}}\}$. The time cost of computing $\gamma(g_i)$ is $O(N_{\rho_i} m^2)$, where m is the number of groups. Hence, the total cost of the basic algorithm is $O(\sum_i (|g_i|^2 N_{avg}^2) + N_{\rho_i} m^2))$ when the aggregates are **sum** and **avg**, and $O(\sum_i (|g_i| N_{max} + N_{\rho_i} m^2))$ when the aggregates are **max** and **min**, where the summations are over all the groups.

4.2 Optimization 1: Pruning

The basic algorithm first computes $\gamma(g_1), \gamma(g_2), \dots, \gamma(g_m)$, and then selects the K largest. In this section, we present pruning method which can decide whether a group g_i is in the top K largest without computing the exact value of $\gamma(g_i)$ and hence reduces the overall time cost.

We use the following three pruning heuristics:

(H1). For groups $G = \{g_1, g_2, \dots, g_m\}$, each $\gamma(g_i)$ can be computed as follows: g_i is considered as the first group, and each other group is considered successively in an arbitrary order. For symbolic simplicity, we write g_i as g_1 . We first consider g_1 and g_2 , compute the PRF value of g_1 , denoted as $\gamma^{(2)}(g_1)$; then add g_3 , compute the PRF value of g_1 , denoted as $\gamma^{(3)}(g_1)$; and so on, until g_m is added, we compute the PRF value of g_1 , denoted as $\gamma^{(m)}(g_1)$. Note that $\gamma(g_1) = \gamma^{(m)}(g_1)$.

When only g_1 and g_2 are considered,

$$\begin{aligned}
\gamma^{(2)}(g_1) &= \omega_1 P(r(g_1) = 1) + \omega_2 P(r(g_1) = 2) \\
&= \omega_1 \int_{-\infty}^{+\infty} \rho_1(x) \int_{-\infty}^x \rho_2(t) dt dx + \omega_2 \int_{-\infty}^{+\infty} \rho_1(x) (1 - \int_{-\infty}^x \rho_2(t) dt) dx \\
&= \omega_1 \int_{-\infty}^{+\infty} \mathcal{F}_1(x) dx + \omega_2 \int_{-\infty}^{+\infty} \mathcal{F}_2(x) dx
\end{aligned} \tag{11}$$

Here, $\mathcal{F}_1(x) = \rho_1(x) \int_{-\infty}^x \rho_2(t)dt$ denotes the multiplication of the aggregate pdf of g_1 and the probability that the aggregate of g_2 is less than x ; $\mathcal{F}_2(x) = \rho_1(x)(1 - \int_{-\infty}^x \rho_2(t)dt)$ denotes the multiplication of the aggregate pdf of g_1 and the probability that the aggregate of g_2 is greater than x .

Now, suppose when g_1, g_2, \dots, g_u are considered, we have computed $\gamma^{(u)}(g_1)$ as

$$\gamma^{(u)}(g_1) = \sum_{i=1}^u \omega_i P(r(g_1) = i) = \sum_{i=1}^u \omega_i \int_{-\infty}^{+\infty} \mathcal{F}_i(x) dx \quad (12)$$

When g_{u+1} is added, $\gamma^{(u+1)}(g_1)$ can be computed as

$$\begin{aligned} & \gamma^{(u+1)}(g_1) \\ &= \sum_{i=1}^u (\omega_i \int_{-\infty}^{+\infty} \mathcal{F}_i(x) \int_{-\infty}^x \rho_{u+1}(t) dt dx + \omega_{i+1} \int_{-\infty}^{+\infty} \mathcal{F}_i(x) (1 - \int_{-\infty}^x \rho_{u+1}(t) dt) dx) \end{aligned} \quad (13)$$

From (13), we can further get

$$\gamma^{(u+1)}(g_1) = \gamma^{(u)}(g_1) + \sum_{i=1}^u (\omega_{i+1} - \omega_i) \int_{-\infty}^{+\infty} \mathcal{F}_i(x) (1 - \int_{-\infty}^x \rho_{u+1}(t) dt) dx \leq \gamma^{(u)}(g_1) \quad (14)$$

Since $\gamma^{(u+1)}(g_1) \leq \gamma^{(u)}(g_1)$ for each u , $2 \leq u \leq m-1$, we can get that $\gamma(g_1) = \gamma^{(m)}(g_1) \leq \gamma^{(u)}(g_1)$. This leads to the following pruning algorithm: we first compute the exact value of $\gamma(g_1), \gamma(g_2), \dots, \gamma(g_K)$ and maintain the top K largest groups retrieved so far. We use L to store the smallest PRF value of the top K largest groups retrieved so far. For each other group g_i , we compute $\gamma^{(u)}(g_i)$ from $u = 2$ to $u = m$ successively. If $\gamma^{(u)}(g_i) < L$ for some u , then g_i cannot be in the top K aggregates and further computations are avoided. Otherwise, we compute the exact value of $\gamma(g_i)$, and insert it into the top- K largest groups, and the group with the smallest PRF value in the original top- K sequence is thrown away.

(H2). For a group g_i , to compute $\gamma(g_i)$, we can divide other groups $G - \{g_i\}$ into three categories: (i) groups $G_1 = \{g' \in G | L_{supp(\rho')} > R_{supp(\rho_i)}\}$ where ρ' denotes the aggregate pdf of g' and $L_{supp(\rho')}$ denotes the left end point of the support of ρ' and $R_{supp(\rho_i)}$ denotes the right end point of the support of ρ_i ; (ii) groups $G_2 = \{g'' \in G | supp(\rho'') \cap supp(\rho_i) \neq \emptyset\}$ where ρ'' denotes the aggregate pdf of g'' and $supp(\rho'')$ and $supp(\rho_i)$ denote the supports of ρ'' and ρ_i respectively, and \emptyset denotes the empty set; (iii) groups $G_3 = \{g''' \in G | R_{supp(\rho''')} < L_{supp(\rho_i)}\}$ where ρ''' denotes the aggregate pdf of g''' and $R_{supp(\rho'''')}$ denotes the right end point of the support of ρ''' and $L_{supp(\rho_i)}$ denotes the left end point of the support of ρ_i . When computing $\gamma(g_i)$, only the groups in G_2 need to be considered. We use $r'(g_i)$ to denote the rank of g_i in G_2 . After $P(r'(g_i) = 1), P(r'(g_i) = 2), \dots, P(r'(g_i) = |G_2|)$ have been computed based on the groups in G_2 , then $P(r(g_i) = 1) = 0, \dots, P(r(g_i) = |G_1|) = 0, P(r(g_i) = |G_1| + 1) = P(r'(g_i) = 1)$,

$\dots, P(r(g_i) = |G_1| + |G_2|) = P(r'(g_i) = |G_2|), P(r(g_i) = |G_1| + |G_2| + 1) = 0, \dots, P(r(g_i) = m) = 0$. The use of this pruning rule will reduce the time cost of computing $\gamma(g_i)$ from $O(N\rho_i|G|^2)$ to $O(N\rho_i|G_2|^2)$.

When computing $\gamma(g_i)$, we can combine (H1) and (H2), only consider the groups in G_2 , and replace $\omega_1, \omega_2, \dots, \omega_u$ with $\omega_{|G_1|+1}, \omega_{|G_1|+2}, \dots, \omega_{|G_1|+u}$ in the corresponding computing equations (11), (12), (13) and (14) of (H1).

(H3). For two groups g' and g'' in G , we use ρ' and ρ'' to denote their aggregate pdfs respectively. If the right end point of the support of ρ' is less than the left end point of ρ'' , then g' must be ranked after g'' . Hence, for a group g in G , if the support of its aggregate pdf has a right end point less than the K th smallest left end point of the supports of the aggregate pdfs of the groups in G , then g must not be in the top K largest groups and can be neglected. After finding the K th smallest left end point α of the aggregate pdfs of the groups in G , the time cost of computing $\gamma(g_i)$ can be avoided if the overall support of ρ_i falls to the left of α .

Note that when applying (H3), if we can decide that group g_i is not in the top- K only according to its aggregate pdf support which can be computed as illustrated in Table 1, then the computing of the aggregate pdf ρ_i can be avoided and the corresponding time cost is saved.

Based on the above three heuristics, we develop our pruning as follows: We first find the K th smallest left end point α of the aggregate pdf supports of the groups in G , and neglect all the groups whose overall aggregate pdf supports fall to the left of α . All the remaining groups are sorted according to their left end points of aggregate pdf supports, and stored in a list \mathcal{L} . Next, we construct a heap \mathcal{H} from the first K groups in \mathcal{L} according to their PRF values. Then, we retrieve a new group g_{new} from \mathcal{L} and compute its upper bound by using heuristics H1 and H2. If the upper bound is less than the lowest PRF value in \mathcal{H} , then g_{new} is not a top K largest group, and the computing of $\gamma(g_{new})$ is stopped. Otherwise, the computing process in heuristic H1 and H2 is continued until we can decide that g_{new} is not a top K largest group or the exact value of $\gamma(g_{new})$ is computed. If $\gamma(g_{new})$ is larger than the lowest PRF value in \mathcal{H} , the group with the lowest PRF value is deleted from \mathcal{H} and g_{new} is inserted into \mathcal{H} . This process is repeated until all the groups in \mathcal{L} have been processed. Finally, all the groups in \mathcal{H} are sorted and returned to the user.

4.3 Optimization 2: Adaptive Strategy

In the basic algorithm and the pruning based optimization discussed until now, a user specified small positive real number δ is used to divide the integral interval into subintervals when computing the corresponding integrals. Smaller δ will make the algorithm more accurate and more time consuming; larger δ will make the algorithm less accurate and less time consuming. We further take an adaptive strategy which can choose appropriate value for δ automatically. We first estimate each $\gamma(g_i)$ as $[lower_i, upper_i]$ by adopting an initial larger δ (the largest support of all the tuple pdfs and aggregate pdfs) in the process of computing the integrals, and decide whether it is in the top- K list. For group g_i , we have three cases: (i) if

$upper_i < lower_{Kth}$, where $lower_{Kth}$ is the lower bound for the K th largest group retrieved so far, then group g_i must not be in the top- K list and can be neglected; (ii) if $lower_i > upper_{Kth}$, where $upper_{Kth}$ is the upper bound for the K th largest group retrieved so far, then the original K th largest group must not be in the top- K list and can be neglected, and group g_i is inserted into the top- K largest groups retrieved so far; (iii) the estimated range for $\gamma(g_i)$ and the K th largest group overlap, we cannot decide whether g_i should be in the top- K list. To insert g_i into the correct position in top- K for case (ii), or to distinguish g_i from the K th largest group in case (iii), the interval which produces the largest error in the estimation of $\gamma(g_i)$ is divided into two subintervals and the corresponding integrals are recomputed in that corresponding intervals and a more accurate estimate for $\gamma(g_i)$ is produced. This process is repeated until g_i is inserted into the correct position or can be distinguished from the K th largest group.

Note that in the process of the estimate of $\gamma(g_i)$ as $[lower_i, upper_i]$, we adopt the adaptive Simpson method [9]

$$\epsilon \leq |(S(a, c) + S(c, b) - S(a, b))/15| \quad (15)$$

where $[a, b]$ is an interval with midpoint c , ϵ is the computing error for the integral on $[a, b]$, and $S(a, b)$, $S(a, c)$, $S(c, b)$ are the estimates given by Simpson's rule on the corresponding intervals. In Equation (12), if $\int_{-\infty}^{+\infty} \mathcal{F}_1(x)dx$, $\int_{-\infty}^{+\infty} \mathcal{F}_2(x)dx$, ..., $\int_{-\infty}^{+\infty} \mathcal{F}_u(x)dx$ have been estimated as $[A_1, B_1]$, $[A_2, B_2]$, ..., $[A_u, B_u]$ respectively, then $\gamma^{(u)}(g_i)$ can be estimated as $[\sum_{j=1}^u \omega_j A_j, \sum_{j=1}^u \omega_j B_j]$. $\int_{-\infty}^{+\infty} \mathcal{F}_1(x)dx$, $\int_{-\infty}^{+\infty} \mathcal{F}_2(x)dx$, ..., $\int_{-\infty}^{+\infty} \mathcal{F}_u(x)dx$ can be estimated recursively according to Equations (7) and (8).

5 Experiments

In this section, we present the experimental evaluation of the basic algorithm and optimizations for computing top- K aggregate queries on continuous uncertain data. Our experiments are conducted on a PC with double 2.20 GHz CPUs and 2GB RAM. The following five algorithms (abbreviated as (G), (B), (P), (A), (PA) respectively) are implemented in C++:

(G): first computing all the group aggregates and then computing all the *PRF* values by using the *generating functions* framework proposed in [7];

(B): the basic algorithm presented in Section 4.1;

(P): the algorithm after combining the pruning heuristics presented in 4.2;

(A): the algorithm after combining the adaptive strategy presented in 4.3;

(PA): the algorithm after combining the pruning heuristics and the adaptive strategy.

For algorithm (G), (B), and (P), we use the smallest δ produced by the adaptive strategy (A). The weights we use for computing the *PRF* values are $\omega_i = 1/i$ ($1 \leq i \leq m$).

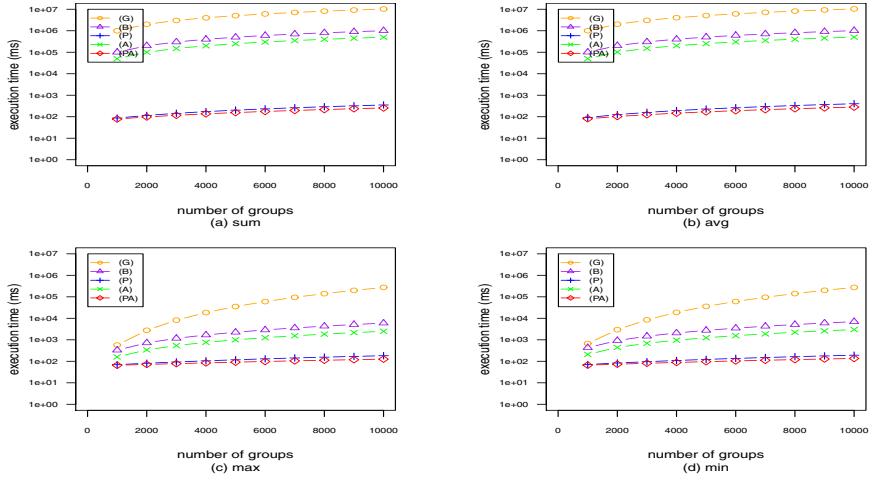


Fig. 2. Running times with respect to different number of groups

5.1 Datasets

We mainly use several synthetic datasets with various continuous distributions to study our algorithms.

Dataset-1-m-c: The aim of this dataset is to investigate the time costs of the various algorithms on the randomly generated dataset with continuous uncertainty. We synthesize a dataset containing 10000 groups with 100 tuples in each group. For each tuple, its score pdf is randomly selected from the four commonly used distributions: Uniform Distribution, Normal Distribution, Gamma Distribution and Beta distribution.

Dataset-2-m-r: The aim of this dataset is to investigate the time costs of the various algorithms on the continuous data set with different maximum group aggregate pdf support overlap rates (the ratio of the maximum number of group aggregate pdfs which have overlapped supports with respect to the overall number of group aggregate pdfs, abbreviated as **MGAPSOR**). We synthesize 50 datasets each of which contains $m (= 1000, 2000, \dots, 10000)$ group aggregate pdfs with 5 different **MGAPSORs** $r (= 10\%, 30\%, 50\%, 70\%, 90\%)$. The group aggregate pdfs are randomly selected from the commonly used Uniform Distribution, Normal Distribution, Gamma Distribution and Beta distribution. Since the **MGAPSOR** does not affect the time costs of the part of computing aggregate pdfs, we assume the aggregate pdfs for all the groups have been computed and synthesize them here for comparing the part of computing group *PRF* values. This is convenient for us to control the **MGAPSOR**.

Note that when a distribution is selected, its parameters (for example, the mean and variance of the normal distribution) are all generated uniformly from $[0, 1000]$. The supports of the Uniform Distribution and the Beta Distribution are bounded naturally. The supports of all the Normal Distributions are truncated

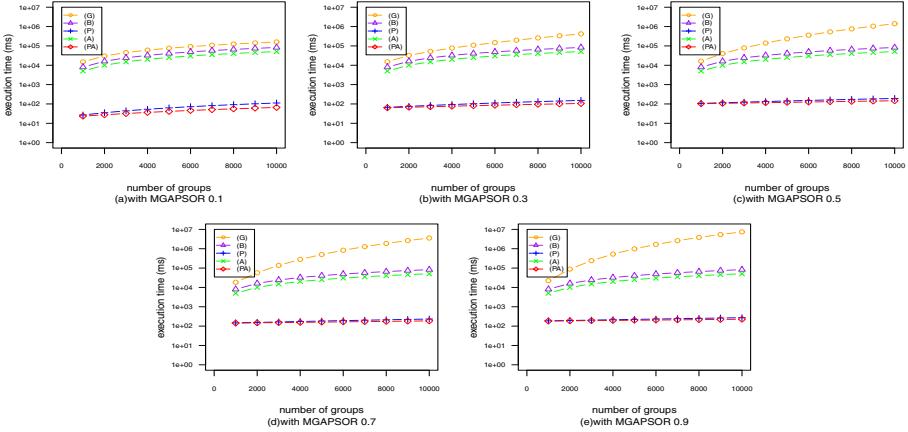


Fig. 3. Running times with respect to different MGAPSOR

to $S = [\mu - r, \mu + r]$ such that $P(S) = 0.9999$, and the supports of all the Gamma Distributions are truncated to $S = [0, r]$ such that $P(S) = 0.9999$.

5.2 Results

In the figures included in this section, each graph shows the running times of the following five algorithms: (G), (B), (P), (A) and (PA). The y axis adopts the logarithmic coordinates.

Figure 2 shows the running times of the algorithms with respect to different number of groups. Graphs (a), (b), (c) and (d), correspond to the **sum**, **avg**, **max**, and **min** aggregate respectively. We set $K = 20$. Each group has a fixed number of 100 tuples. We see that the time cost of Algorithm (G) is higher than Algorithm (B). The algorithm (PA) has a much lower time cost than the basic algorithm (B). As the number of groups increases, the pruning based optimization and the adaptive strategy are more effective for computing the top- K aggregate query than the basic algorithm.

We also run the algorithms on the datasets with different number of tuples in each group, and with different number of K . We all get the same conclusion that on our randomly generated datasets with continuous uncertainty, the basic algorithm (B) has a lower time cost than algorithm (G), and the algorithm after combining the pruning heuristics and the adaptive strategy (PA) has a much lower time cost than the basic algorithm (B).

Figure 3 shows the running times of the algorithms on Dataset-2-m-r. Graphs (a), (b), (c), (d), and (e) correspond to the **MGAPSOR** 0.1, 0.3, 0.5, 0.7, 0.9 respectively. We see that the time cost of Algorithm (B) is lower than Algorithm (G) when m increases to some fixed number. After adopting the pruning method and the adaptive strategy (PA), the time cost is lower than (G) and (B) for all m . For higher **MGAPSOR**, the pruning method and the adaptive strategy (PA) are more effective than (G) and (B).

6 Conclusions

We formally formulated the top- K aggregate queries on continuous uncertain data, and developed computing algorithms and optimizations (pruning heuristics and adaptive strategy). We mainly focus on continuous attribute uncertainty in this paper and assume that all the attribute values are independent of each other. The extension to support tuple uncertainty and incorporating correlations are left as future work. Applying our algorithms to real world applications is also another important task for future work.

Acknowledgement. The work is supported by National Natural Science Foundation of China (60773156, 61073004), Chinese Major State Basic Research Development 973 Program (2011CB302203-2), Important National Science & Technology Specific Program (2011ZX01042-001-002-2), and research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

1. Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: A system for data, uncertainty, and lineage. In: VLDB (2006)
2. Cheng, R., Kalahnikov, D.V., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: SIGMOD (2003)
3. Dalvi, N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB Journal 16(4) (2007)
4. Ge, T., Zdonik, S., Madden, S.: Top- k queries on uncertain data: On score distribution and typical answers. In: SIGMOD (2009)
5. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: A probabilistic threshold approach. In: SIGMOD (2008)
6. Jestes, J., Cormode, G., Li, F., Yi, K.: Semantics of ranking queries for probabilistic data. TKDE (2011)
7. Li, J., Deshpande, A.: Ranking continuous probabilistic datasets. In: VLDB (2010)
8. Lian, X., Chen, L.: Probabilistic inverse ranking queries in uncertain databases. The VLDB Journal (2011)
9. Lyness, J.N.: Notes on the adaptive simpson quadrature routine. Journal of ACM (1969)
10. Ré, C., Dalvi, N., Suciu, D.: Efficient top- k query evaluation on probabilistic data. In: ICDE (2007)
11. Soliman, M.A., Ilyas, I.F.: Probabilistic top- k and ranking-aggregate queries. TODS (2008)
12. Soliman, M.A., Ilyas, I.F.: Ranking with uncertain scores. In: ICDE (2009)
13. Soliman, M.A., Ilyas, I.F., Chang, K.C.-C.: Top- k query processing in uncertain databases. In: ICDE (2007)
14. Wang, C., Yuan, L.Y., You, H.-H., Zaiane, O.R.: On pruning for top- k ranking in uncertain databases. In: VLDB (2011)
15. Lian, X., Chen, L.: Probabilistic ranked queries in uncertain databases. In: EDBT (2008)
16. Yi, K., Li, F., Kollios, G., Srivastava, D.: Efficient processing of top- k queries in uncertain databases with x-relations. TKDE (2009)

Shortest Path Finder with Light Materialized Path View for Location Based Services

Aye Thida Hlaing¹, Htoo Htoo¹, Yutaka Ohsawa¹,
Noboru Sonehara², and Masao Sakauchi²

¹ Graduate School of Science and Engineering, Saitama University

² National Institute of Informatics

Abstract. Fast shortest path search between two points on a road network is essential demand in location based services (LBS). For this purpose, several types of road network distance materialization methods have been studied. The distance materialization approach is quite fast, however, it results in a huge amount of data. This paper proposes a shortest path search algorithm based on materialized-path-view constructed only on partitioned subgraphs, and its three variations referring different levels of distance materialization. The amount of pre-computed data is greatly reduced. The shortest path is retrieved by a best-first-search using a priority queue. The difference between three variations of the algorithm is the materialization level of the distance in the subgraphs. The performance of them is evaluated comparing with A* algorithm and HEPV experimentally. Through the results, we show the proposed algorithm outperforms the conventional methods.

1 Introduction

Point of interest (POI) queries based on the road network distance become an important role on location based services (LBS). For these queries, the optimization on the distance or the time of travel along the road network is important besides the Euclidean distance.

Several methods based on materialized path view (MPV) have also been proposed for the fast road network distance computation. They retrieve the distance by looking up a pre-computed distance table. However, this MPV has the following problems: (1) Usually, a road network contains a large amount of nodes, and the data size of the MPV is proportional to the square of the number of nodes. Therefore, the data amount of the distance table becomes huge for a large size of the road network. (2) Very long processing time is necessary to construct MPV table, because the distance must be calculated over all combinations of node pairs. As concerns to the data amount of the table, when the total number of nodes in a graph is 1,000,000 (it corresponds to a road network over the range about 100km square), the number of elements in MPV table becomes 10^{12} , therefore several TB memory is required. (3) When the weight values (e.g. length) of some links in the network are changed by a traffic accident or a construction,

these changes affect the wide area on the table. This update also requires a long processing time.

To cope with these problems, hierarchical MPV methods have been proposed [1][2]. These methods alleviate the problems described above, however, the problems cannot be avoided authentically. A change in a leaf level affects to the upper levels. Long computation time is necessary for the upper level distance calculation. The data amount in a high level layer is not always smaller than that of the leaf level, in opposition to a usual hierarchical tree structure.

In a query for LBS, the shortest path must be determined from a large number of candidates, and the area where candidates exist is limited in a confined area, for example, searches in an area having 50km radius centered the query point.

This paper proposes a shortest path search algorithm based on a lightweight local distance materialization, which is constructed on a partition of a road network. These methods outperform A* algorithm, and they reduce the data amount drastically comparing with the conventional hierarchical distance materialization methods.

2 Shortest Path Finder

2.1 Data Structure

A road network is modeled as a directed graph $G(V, E, W)$, where V is a set of nodes (intersections), E is the set of edges (road segments), and W is the set of link weights. A fragment $SG_i(V_i, E_i, W_i)$ of a graph $G(V, E, W)$ is a partitioned subgraph, where $V_i \in V$, $E_i \in E$, and $W_i \in W$. If the end points of an edge $e_{jk} \in E_i$ are v_j and v_k , then $v_j \in V_i$ and $v_k \in V_i$. This subgraph is denoted as SG_i in the rest of the paper where there is no ambiguity.

Fig. 1(a) shows an example of a road network graph, here small circles are nodes and lines are edges. Fig. 1 (b) depicts a partition of the graph shown Fig. 1(a). In this partition, the nodes shown by black dots belong to at least two neighboring subgraphs; the nodes belonging to the plural subgraphs are called the *border nodes*. Two subgraphs are defined adjacent if they have at least one common border node. The set of border nodes of SG_i is denoted by BV_i . In this partition, each edge belongs to only one subgraph. The nodes shown in white circles in the figure are referred as *inner nodes*: they are the rest of the nodes in a subgraph except the border nodes.

Three types of distance materialization tables are referred in the proposed methods; a border-to-border distance table (BBDT), an inner-to-border distance table (IBDT), and a node-to-node distance table (NNDT). The BBDT shows the distances between two border nodes. The distances in these tables are calculated by traveling inside of the subgraph, therefore these values are not always global shortest path lengths.

The IBDT shows the distance from an inner node to a border node. This table is used to retrieve the distance from the starting point as an inner node to a border node. An NNDT lists distances of all combinations of the nodes in a

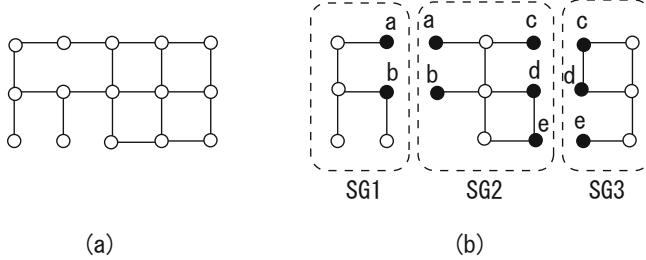


Fig. 1. Flat graph and its partition

subgraph. This table is used to know the distance between two arbitrarily specified nodes. Either IBDT or NNDT is used alternatively in the SPF algorithms described in the next subsection.

2.2 Simple Path Finder Algorithm

Fig. 2 shows the processing flow of the shortest path finder (SPF). In the following description, s and d denote the starting point and the destination point of the shortest path to be retrieved. The SPF is controlled by a best-first search using a priority queue (PQ). The PQ manages the records constructed by the following items.

$$\langle p, Cost, df_s, fSG, phase \rangle$$

Here, p is the currently noticed point; s , d , or a border node. $Cost$ is the lower bound road network distance between s and d . The PQ returns the record by ascending order of this value. df_s (distance-from-source) is the shortest road network distance between s and the currently noticed node p . fSG is the subgraph ID in which p belongs. The last item, $phase$ is a value to show the progress of the processing. It is changed from PHASE0 (initial state) to PHASE3 (final state) according to the progress of the processing.

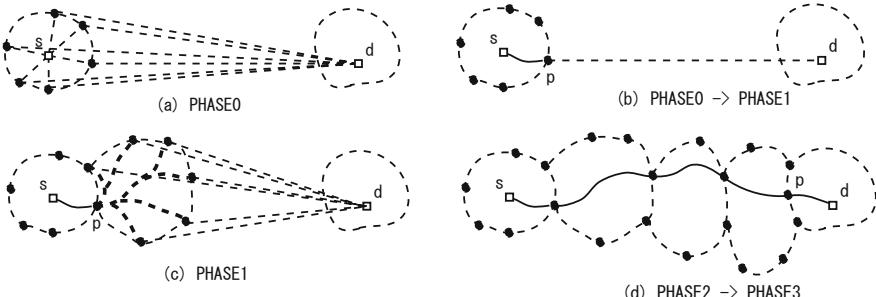


Fig. 2. Processing flow of SPF

At first, the subgraph, SG_s , which contains the road segment under s , is determined. Next, $Cost$ is calculated by the equation, $Cost = d_E(s, b_i) + d_E(b_i, d)$, for all border nodes $b_i \in BV_s$ of SG_s . Here, $d_E(x, y)$ denotes the Euclidean distance between x and y . In this initial stage, the following records are composed and enqueue to the PQ. In this processing stage, the records have $PHASE0$ as the $phase$ value.

$$< b_i, d_E(s, b_i) + d_E(b_i, d), 0, SG_s, PHASE0 > \text{ for all } b_i \in BV_s$$

Next, a record (e) that has minimum $Cost$ value is dequeued from the PQ as shown in Fig. 2(b). At the beginning of the processing, $e.phase$ is $PHASE0$. For the border node $e.p$, the road network distance $d_N(s, e.p)$ is calculated. Here, $d_N(x, y)$ denotes the road network distance between x and y . $Cost$ value for this node is calculated by the equation $Cost = d_N(s, e.p) + d_E(e.p, d)$, composing the following record, and then it is enqueue in the PQ.

$$< e.p, Cost, d_N(s, e.p), e.fSG, PHASE1 >$$

When the $phase$ value of the obtained record (e) from the PQ is $PHASE1$ (see Fig. 2(c)), the road network distance from s to the current node ($e.p$) has already been determined. All subgraphs that contain $e.p$ as a border node are also determined. And then, for each subgraph SG_n , $Cost$ is calculated by the following equation.

$$Cost = e.dfs + d_N(p, b_i) + d_E(b_i, d) \quad (b_i \in BV_n)$$

Here, BV_n is a border node set of SG_n . The following record is composed, and then it is enqueue in the PQ.

$$< b_i, Cost, e.dfs + d_N(p, b_i), SG_n, PHASE1 >$$

Continuing the processing, when a record obtained from the PQ reaches a border node of the subgraph containing d , the record shown below is composed and it is enqueue in the PQ.

$$< e.p, e.dfs + d_E(e.p, d), e.dfs, e.fSG, PHASE2 >$$

In this case, the value of the road network distance from s to $e.p$ plus the Euclidean distance between $e.p$ and d is assigned to $Cost$ value, and $PHASE2$ is assigned to $phase$ value.

When the $phase$ value of the dequeued record from the PQ is $PHASE2$, the road network distance between $e.p$ and d is calculated. Composing the following record, and it is enqueue into the PQ.

$$< e.p, e.dfs + d_N(e.p, d), e.dfs, SG_d, PHASE3 >$$

When the $phase$ value of the dequeued record is $PHASE3$, the shortest path distance between s and d has been determined. The fact that the record is dequeued from the PQ means it has the minimum $Cost$ value among all records contained in the PQ. It means the shortest path distance is determined. Therefore, return the distance, and then the searching process is terminated.

3 Experimental Results

We evaluated three variations of SPF; SPFLM (SPF with light materialization), SPFMM (SPF with medium materialization), and SPFFM (SPF with full materialization); with two representative conventional methods, PWA* algorithms and HEPV [1]. The difference between these methods is how to determine the distance between one of the specified points (s and d) and the border nodes of the subgraph where the point belongs to. SPFLM calculates the distance by A* algorithm referring usual adjacency list. SPFMM obtains the distance by referring the IBDT. SPFFM determines the distance by referring the NNDT.

Table 1 shows the tables sizes of three road network maps used in this experiment.

Table 1. Data size (MB)

map name	PWA*	SPFLM	SPFMM	SPFFM	HEPV	Next Hop
MapS	1.5	2.6	6.7	14.5	30.1	13.4
MapM	6.8	11.3	28.7	70.1	376.1	65.6
MapL	39.7	65.8	166.6	400.0	8,287.6	373.8

Fig. 3(a) compares the processing time of the shortest path searching among the PWA*, SPFLM, SPFFM, and two layered HEPV, using MapS divided into 100 subgraphs. The horizontal axis shows the distance between s and d . We generated 1,000 pairs of s and d by a pseudo-random sequence. For each $s - d$ pair, the shortest path was searched by five algorithms. (SPFMM is omitted from this figure to avoid intricacy: it performed almost the same as SPFFM.) This figure presents the results that is selected one after every 5 queries. All LRU buffers were cleared in advance for every query. The most of processing time by SPFLM, SPFFM, and HEPV stay under 20 ms over the whole distance range. Meanwhile, the processing time of PWA* increases almost linearly in accordance with the increase of the distance.

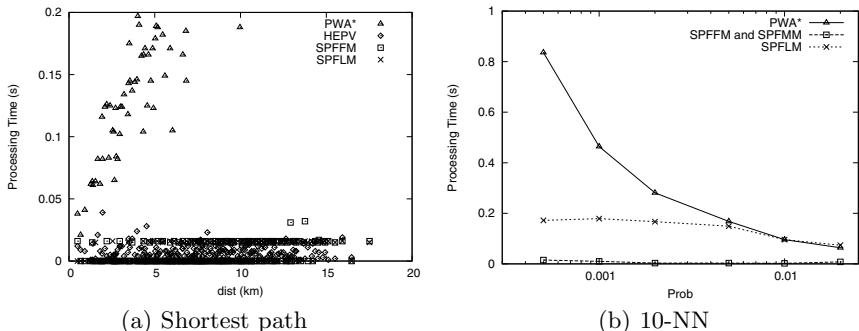


Fig. 3. Processing time for shortest path searches and 10-NN searches

Next, we generate several sets of points by a pseudo-random sequence to simulate the points of interest (POI) on the road network links. The number of generated points were specified by a probability $Prob$. For example, when $Prob = 0.01$, a POI exists on 100 road links. We searched 10 nearest neighbor (NN) POIs of a query point (q) in Euclidian distance. After that, the road network distances are computed for the found POIs by PWA*, SPFLM, SPFMM, and SPFFM. We determined 10 query points randomly on the road network. Fig. 3(b) shows the average processing time spent to determine 10 shortest paths.

These three results show similar processing times for the same $Prob$. For denser than $Prob = 0.002$, the processing time of the SPFLM shows almost the same value with PWA*. This is because the path length is small in high $Prob$ values, and PWA* algorithm can run fast.

4 Conclusion

This paper proposes a shortest path search algorithm and its three variations using the light distance materialization that are suitable for LBS. The data amount used in these methods can be reduced in comparing with the conventional hierarchical network distance materialized methods; HEPV and HiTi. Especially, SPFLM reduces the data amount drastically. On the other hand, SPFMM and SPFFM achieve similar time efficiency with the hierarchical encoded path view.

Consequently, when the distance between two points is large, SPFLM outperforms PWA* substantially, nevertheless the SPFLM uses a small amount of pre-computation data. LBS is apt to request for the shortest path searches over rather than nearly located points, and the operation is repeated over a large number of times in a query; for example as in the incremental Euclidean restriction strategy. In this situation, the relative search speed of PWA* increases, because the hit ratio in LRU buffer managing adjacency list is increased. k NN queries evaluated in this paper is for such example. When the density of POI is high, the difference of the processing times between SPFLM and PWA* becomes small. On the other hand, SPFMM and SPFFM outperform the other methods even in such situation.

Acknowledgments. The present study was partially supported by the Japanese Ministry of Education, Science, Sports and Culture (Grant-in-Aid Scientific Research (C) 24500107 and (B) 2300337).

References

1. Jing, N., Huang, Y.W., Rundensteiner, E.A.: Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *IEEE Transactions on Knowledge and Data Engineering* 10(3), 409–432 (1998)
2. Jung, S., Pramanik, S.: An efficient path computation model for hierarchically structured topographical road maps. *IEEE Transactions on Knowledge and Data Engineering* 14(5), 1029–1046 (2002)

RUM+-tree: A New Multidimensional Index Supporting Frequent Updates

Yuean Zhu^{1,2}, Shan Wang^{1,2}, Xuan Zhou^{1,2}, and Yansong Zhang^{1,2,3}

¹ DEKE Lab, Renmin University of China, Beijing 100872, China

² School of Information, Renmin University of China, Beijing 100872, China

³ National Survey Research Center at Renmin University of China, Beijing 100872, China
`{iwillgoon, swang}@ruc.edu.cn`

Abstract. Update-intensive applications, such as location-aware services, sensor networks, and stream databases, face with the problem of frequent updating. R-tree and its variants are dominant indexing structures for multi-dimensional object. An update in R-tree traditionally consists of a deletion and an insertion, which is sometimes inefficient. In this paper, we present an R-tree-based index structure called RUM+-tree for enabling efficient frequent updates. In RUM+-tree, compared to R-tree, two additional data structures are maintained: an Update Memo and a hash table. An object's leaf node can be accessed directly through the hash table and the Update Memo records the versioning information of the most recent data. The idea of RUM+-Tree is to deal with updates locally in a bottom-up manner as often as possible. Only when the update involves multiple MBRs, a multi-version solution is utilized for acceleration. Compared to RUM-tree [1], RUM+-tree can reduce the new version to be created. Thus, it can accelerate the *garbage clean* and update procedure. Compared to LUR-tree [5], RUM+-tree eliminates traditional updates like used by R-tree completely. Our experimental results show that our technique outperforms RUM-tree and LUR-tree significantly.

Keywords: Indexing moving objects, R-trees, frequent update, bottom-up update, multi-version.

1 Introduction

Nowadays, a wide range of applications, including goods delivery and traffic control, need to store and process the continuously changing information of moving objects. Except that, there are also lots of applications relying on the sampling continuous and multidimensional variables. All these applications are characterized by frequent updates. Usually, every sampled data value triggers an update to the underlying database server. So, it is essential to develop spatial indexes to handle large volumes of updates in an efficient manner for those scenarios. The dominant indexing technique for multidimensional data with low dimensionality is R-tree [3] which is originally designed for query-intensive setting, i.e., one of the primary concerns in R-tree is to minimize the search cost of spatial queries. In R-tree, the update operation is very costly since it treats an update as combination of a deletion and an insertion. Given the fact that the

MBRs (Minimal Bounding Rectangle) in an R-tree may overlap, locating the old entry to be deleted may follow multiple paths from the root towards the leaf node and thus is expensive. Hence, it is believed that the original R-tree isn't a good choice for update-intensive setting, where frequent updates are continuously generated.

In this paper, we propose a new R-tree based index structure which efficiently supports frequent updates. As it is an extension of the RUM-tree, we call our method RUM+-tree. RUM+-tree does not perform expensive traditional top-down update in R-tree. Instead, it maintains two data structures in addition to R-tree: a hash table on object ID and update memo. When an update comes, the leaf node of the indexed object can be directly located through the hash table. If the new position of the object still falls in the same MBR as its old position, RUM+-tree just modifies the position of the object in the corresponding leaf node and the update is finished. Otherwise, utilizing the update memo, the update is turned into an insertion. The application of update memo allows multiple versions of one object to coexist in the same RUM+-tree, and the deletion can conduct periodically in a batch manner. It can thus save a significant proportion of I/O cost. We have carried out extensive experiments to demonstrate the efficiency of RUM+-tree.

The rest of the paper is organized as follows: Section 2 discusses the related work on indexing moving object and supporting frequent updates for R-trees. In section 3, we present our approach. Section 4 presents the results of our performance evaluation, in which we compared RUM+-tree against several previous approaches. Finally, we conclude in section 5.

2 Related Works

In the last two decades, with the recent attention on indexing moving objects, many index structures have been proposed to index moving object. According to the type of data being stored, these data structures are divided into two categories: (1) indexing the historical trajectories of objects [4] (2) indexing current and predicted location of moving objects [1, 2, 5, 8]. All these methods must handle frequent updates. However, Traditional update algorithm in R-tree is inefficient. To support frequent updates in R-trees, a bottom-up update style is proposed in the [2, 5]. These approaches are based on the observation that some updates can be handled locally. In [2], by using a secondary index, the leaf node which contains the object involved in update query can be directly accessed. If the consecutive changes of the object are small, the update can be finished quickly. [5] polishes this idea by maintaining an in-memory summary structure to help both updates and queries. The bottom-up approach works well when the consecutive changes of objects are still in the same MBR bounding. However, when consecutive changes are large, their performance will be affected greatly. In the worst case, its performance is the same with traditional update, i.e., bottom-up update is no more applicable. On the other hand, to eliminate the traditional update adopted by R-tree, [1] proposed a memo-based updating method, called RUM-tree. When an update comes, it is assigned a timestamp. Then, along with the object ID and timestamp, new value is inserted into the tree. RUM-tree defers the deletion and the deletion is performed in a batch manner after the number of updates handled exceeds a threshold.

3 RUM+-tree

In this section, we present a novel index structure called RUM+-tree to support frequent update in the context of indexing current position of moving object. First, we introduce the basic idea of our approach. Then, we describe the structure of RUM+-tree.

3.1 Basic Idea

We use the scenario of figure 1 to illustrate the disadvantage of RUM-tree. R_1 and R_2 are MBRs on leaf nodes. R_0 is the MBR of the parent node of R_1 and R_2 . obj_1 belongs to R_2 at the beginning. Assume the fanout of entries in the leaf node is 3. R_1 has 3 entries and R_2 has 3 entries as well (including obj_1). Triangles represent objects belonging to R_1 and circles represent belonging to R_2 . At some point, the movement of obj_1 triggers an update. Then, RUM-tree prepares to create new version like in the figure 1(b). The leaf node is full, and this causes a split. As shown in figure 1(c), after creating new version for obj_1 , Since R_2 has reached the maximum cardinality of node, there is no room for the new version of obj_1 . R_2 is split into R_{2a} and R_{2b} . As we can see, continuous creation of new version in the same leaf node has some drawbacks. One drawback is that it will cause node split. What is worse is that the split can propagate upwards. Second, when the *garbage clean* is triggered to remove the old versions, the node may be merged again. This split and merge procedure is expensive and unnecessary. To avoid it, we equip the RUM-tree with the lazy update feature. When an update is coming, we check if the update is local. If so, only the leaf node affected by the update is modified and the update is finished. Otherwise, we are sure that the consecutive change for the object is large, and the insertion may be uniform among the leaf nodes, i.e., the insertions may not always happen in the same leaf node. Back to the above example, our algorithm does not have to handle the update figure 1(b). Because new position of an object is still in the same MBR to which the object belongs, we do not have to create new version for obj_1 . What that has to be done is to update the position of the corresponding entry in the leaf node. In this case, we need only 2 disk accesses (one read R_1 and one write R_2) and reduce the cost by at least 2 I/Os. On the one hand, by reducing the number of new versions to be created, it can accelerate the *garbage clean* and update procedure. Additionally, Compared to LUR-tree, RUM+-tree eliminates traditional update like in R-tree completely.

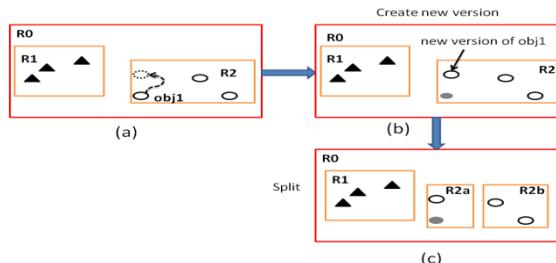


Fig. 1. One possible result caused by RUM-tree in tackling update

3.2 Index Structure

Based on the above observation, we propose a new index structure, called RUM+-tree. Compared to R-tree, RUM+-tree is equipped with two additional components: a secondary index and an update memo. The secondary index enables directly access to the leaf node of each object; the update memo provides two functions: to identify the latest version of the data and to keep track of the old versions. The update memo also plays an important role in finding search result. In RUM+-tree, the raw search result is a superset of the actual answer. Once the raw search result is obtained, it has to be compared with the update memo to eliminate the false answers. In RUM+-tree, the MBR of leaf node is computed on the fly. If the object does not move out of the MBR, the update is done by modifying the position of the object. Otherwise, together with timestamp and the object ID, the new value is inserted into RUM+-tree. RUM+-tree doesn't sacrifice fanout of the tree and thus sustains the search performance. The structure of RUM+-tree is shown in figure 2.

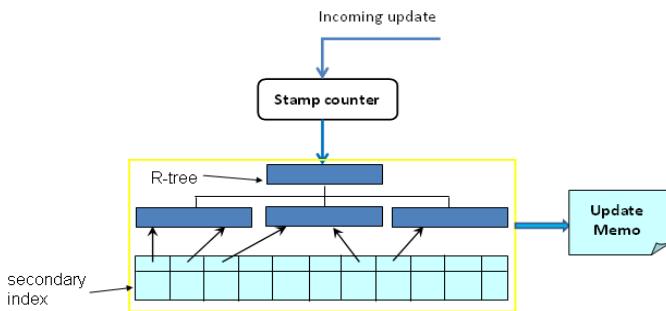


Fig. 2. Structure of RUM+-tree

4 Performance Evaluation

In this section, we will show the results of experiments and analyze the performance of the RUM+-tree regarding to the update performance. To evaluate the proposed RUM+-tree, RUM+-tree is compared with the conventional R*-tree, LUR-tree(Lazy Update R-tree) and RUM-tree(R-tree with Update Memos)in terms of the number of page accesses. All the experiments are carried on Intel Pentium Dual-Core T2390 1.86GHz PC with 2G RAM. We implement the RUM+-tree on the basis of R*-tree [11]. We use Beckmann's R*-tree source code. The development environment is VS 2005 and all algorithms are implemented using c. The size of disk page is 4KB. The R*-tree update algorithm is made of deletion and insertion. The data sets in the experiment are generated by GSTD [6]. We generate four data sets with different initial distribution and different movement pattern of object. The initial distribution and movement pattern are U-R (Uniform, Random), U-D (uniform, directed), G-R (Gaussian, Random) and G-D (Gaussian, Directed) respectively. There are no objects disappearing or no new objects appearing in the experiment. So, the secondary index can be implemented as static hash table without overflow. The working space of the objects is the unit square. We keep the secondary index in main-memory.

4.1 Update Performance

The number of update operations ranges from 15,000 to 150,000 and the number of moving object ranges from 1,000 to 10,000. Figure 3 gives the average I/Os for all evaluated algorithms. The average I/Os in RUM-tree are nearly stable. The reason is that the RUM-tree treats the update operation as insertion and the I/O cost of insertion algorithm is only related with the height of tree. If the overlaps of the MBR is serious, the update performance of R*-tree will deteriorate greatly. That's why there are more jumps in the performance illustration of R*-tree. RUM+-tree behaves well in update operation and is less affected by objects' movement pattern compared with LRU-tree. In the worst case, i.e., no update can be done locally, the LRU-tree is the same with R*-tree and RUM+-tree is the same with RUM-tree.

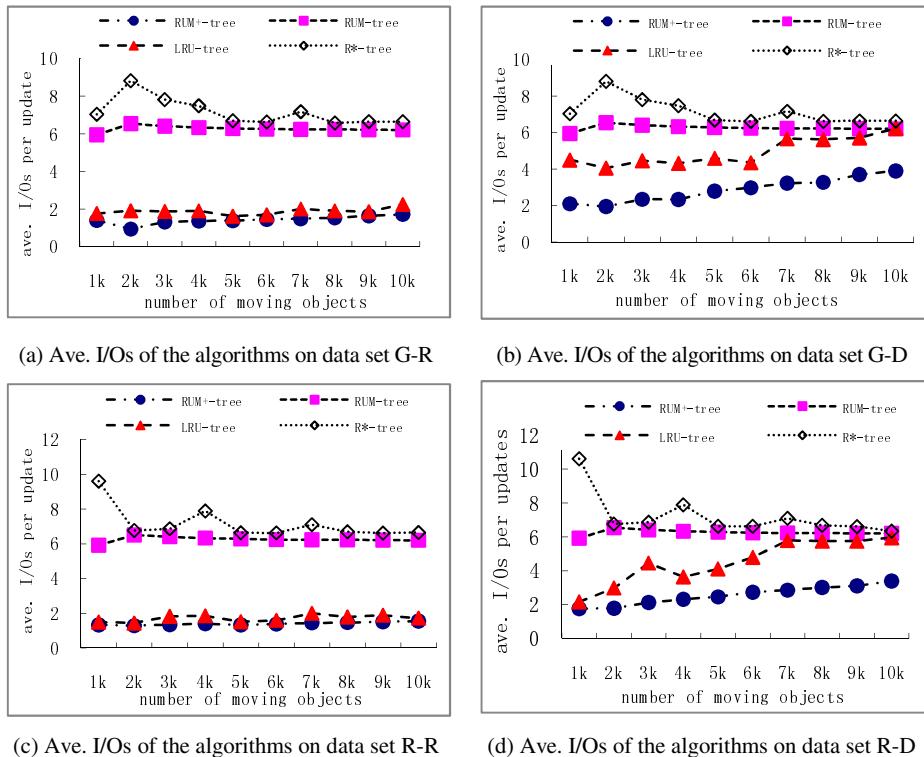


Fig. 3. Average I/Os of the evaluated algorithms on all data sets

5 Conclusion and Future Work

In this paper, we propose a new R-tree based index, called RUM+-tree, to support the intensive update. We improve the RUM-tree by equipping it with lazy update feature to reduce production of version data. Experiments show that our approach is efficient. In some case, RUM+-tree can greatly improve the update performance

compared with RUM-tree. The pure bottom-up update manner, such as used by LRU-tree will be greatly affected by the movement pattern of object. In worst case, it is the same with the update in R-tree. The RUM+-tree is much less sensitive to movement pattern of object. Furth more, the RUM+-tree eliminates the traditional update manner like in R-tree completely.

With development of new hardware platform such as multi-core and huge RAM, [7, 9, 10] let the all data reside in the RAM and achieve good response time in supporting huge volume of updates. Our future work is to develop the in-memory version of RUM+-tree or take the memory buffer into consideration.

Acknowledgement. This work has been supported by the NEC Corporation and Graduate Science Foundation of Renmin University of China (No.13XNH216). The author also wants to thanks for the advice from Yubin Guo from SCAU.

References

1. Xiong, X., Aref, W.G.: R-Trees with Update Memos. In: ICDE (2006)
2. Lee, M.-L., et al.: Supporting Frequent Updates in R-Trees: A Bottom-Up Approach. In: VLDB, pp. 608–619 (2003)
3. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: ACM SIGMOD, pp. 47–57 (1984)
4. Pfoser, D., et al.: Novel Approaches in Query Processing for Moving Object Trajectories. In: VLDB, pp. 395–406 (2000)
5. Kwon, D., Lee, S., Lee, S.: Indexing the Current Positions of Moving Objects Using the Lazy Update R-Tree. In: MDM, pp. 113–120 (2002)
6. Theodoridis, Y., Silva, J.R.O., Nascimento, M.A.: On the generation of spatiotemporal datasets. In: Güting, R.H., Papadias, D., Lochovsky, F.H. (eds.) SSD 1999. LNCS, vol. 1651, pp. 147–164. Springer, Heidelberg (1999)
7. Dittrich, J., Blunschi, L., Vaz Salles, M.A.: Indexing moving objects using short-lived throwaway indexes. In: Mamoulis, N., Seidl, T., Pedersen, T.B., Torp, K., Assent, I. (eds.) SSTD 2009. LNCS, vol. 5644, pp. 189–207. Springer, Heidelberg (2009)
8. Nguyen, T., He, Z., Zhang, R., Ward, P.: Boosting moving object indexing through velocity partitioning. In: VLDB, pp. 860–871 (2012)
9. Šidlauskas, D., Ross, K.A., Jensen, C.S., Šaltenis, S.: Thread-Level Parallel Indexing of Update Intensive Moving-Object Workloads. In: Pfoser, D., Tao, Y., Mouratidis, K., Nascimento, M.A., Mokbel, M., Shekhar, S., Huang, Y. (eds.) SSTD 2011. LNCS, vol. 6849, pp. 186–204. Springer, Heidelberg (2011)
10. Šidlauskas, D., Šaltenis, S., Jensen, C.S.: Parallel main-memory indexing for moving-object query and update workloads. In: ACM SIGMOD, pp. 37–48 (2012)
11. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In: ACM SIGMOD, pp. 322–331 (1990)

Joint Clustering and Feature Selection

Liang Du^{1,2,3} and Yi-Dong Shen¹

¹ State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, Beijing, 100190, China
² Graduate University of Chinese Academy of Sciences
³ University of Chinese Academy of Sciences, Beijing, 100049, China
{duliang,ydshen}@ios.ac.cn

Abstract. Due to the absence of class labels, unsupervised feature selection is much more difficult than supervised feature selection. Traditional unsupervised feature selection algorithms usually select features to preserve the structure of the data set. Inspired from the recent developments on discriminative clustering, we propose in this paper a novel unsupervised feature selection approach via Joint Clustering and Feature Selection (JCFS). Specifically, we integrate Fisher score into the clustering framework. We select those features such that the fisher criterion is maximized and the manifold structure can be best preserved simultaneously. We also discover the connection between JCFS and other clustering and feature selection methods, such as discriminative K-means, JELSR and DCS. Experimental results on real world data sets demonstrated the effectiveness of the proposed algorithm.

Keywords: Unsupervised Feature Selection, Fisher Score, Spectral Clustering.

1 Introduction

In many applications in data mining and machine learning, one is often confronted with very high dimensional data. High dimensional data require more time and space to process. Moreover, due to the existence of many irrelevant and/or redundant features, learning algorithms tend to overfit. To overcome this problem, feature selection techniques are designed to select a subset of features from the high dimensional feature set for a compact and accurate data representation.

Feature selection methods can be classified into supervised and unsupervised methods. Supervised feature selection algorithms, such as Fisher score [1], robust sparse regression [2], and trace ratio [3], select features according to the relevance between features and labels. However, in practice, the labels are expensive to obtain. Hence, it is important to develop unsupervised feature selection algorithms by using all the data points without labels.

Recently, several unsupervised algorithms have been proposed to leverage both the manifold structure and learning mechanism. These methods include

Laplacain Score (Lap Socre) [4], Spectral Feature Selection (SPEC) [5], Multi-Cluster Feature Selection (MCFS) [6], Unsupervised Discriminative Feature Selection (UDFS) [7], joint Feature Selection and Subspace Learning (FSSL) [8] and Joint Embedding Learning and Sparse Regression (JELSR) [9]. Commonly, these approaches characterize manifold structure via various graphs and select features with different learning mechanism. LapScore and SPEC rank each features by computing different metrics. MCFS achieves feature selection by Spectral Regression [10]. Both of UDFS, FSSL and JELSR can viewed as an integration of embedding with different graphs and sparse subspace learning via $\ell_{2,1}$ -norm regularization.

Besides, to inherit the discriminative power of supervised approach, several recent work [11–13] incorporate supervised subspace learning technique into the clustering framework. Empirical results have shown performance improvement with other popular clustering algorithms.

Based on the above motivation, in this paper, we propose a new approach, called *Jointly Clustering and Feature Selection* (JCFS), for unsupervised feature selection. Specifically, we integrate supervised feature selection technique (Fisher Score) and spectral clustering (manifold learning) in a unified framework. Based on the selected features, we jointly maximize Fisher criterion for feature selection and minimize the spectral clustering criterion to preserve the manifold structure. Compared with traditional unsupervised feature selection approaches, our method could integrate the merits of supervised feature selection and clustering (manifold learning). We also discover the connection between JCFS and other clustering and feature selection methods, such as spectral clustering, discriminative K-means, JELSR and Discriminative codebook selection [14]. Many experimental results are provided for demonstration.

2 Notations

Suppose we have n data points $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbf{R}^d$, the data matrix is denoted by $X = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbf{R}^{d \times n}$. We use \mathbf{x}^i to represent the i -th feature of X . We represent a clustering of the data points by a partition matrix $P = [\mathbf{p}_1, \dots, \mathbf{p}_c] = [p_{ij}] \in \{0, 1\}^{n \times c}$ and $P\mathbf{1}_c = \mathbf{1}_n$. Thus, exactly one element in each row of P is 1. Instead of directly using the entries of partition matrix P as the cluster labels, we use a Scaled Partition Matrix $Y = P(P^T P)^{-1/2}$. Without loss of generality, we assume that X has been centered with zero mean, i.e., $\sum_i^n \mathbf{x}_i = \mathbf{0}$.

3 A Review of Fisher Score and Spectral Clustering

In this section, we briefly introduce Fisher Score [1] and Spectral Clustering [15].

3.1 Fisher Score for Feature Selection

The key idea of Fisher score [1] is to find a subset of features, based on theses selected features the distances between data points in different classes are as

large as possible, while the distances between data points in the same class are as small as possible. The *Fisher Score* can be formulated as the following *Ratio Trace* optimization problem:

$$\max_{X^m} \text{tr}(S_t + \gamma I_{d \times d})^{-1} S_b \quad (1)$$

where $\text{tr}(\cdot)$ is the trace of a squared matrix, X^m is the data matrix represented by m selected features, the total scatter matrix S_t and the between-cluster scatter matrix S_b are defined as follows [12]:

$$S_t = X^m (X^m)^T \quad S_b = X^m Y Y^T (X^m)^T \quad (2)$$

The regularization parameter $\gamma > 0$ in (1) is used to keep the total scatter matrix non-singular. To represent whether a feature is selected or not, we introduce an indicator variable $\mathbf{z} = [z_1, \dots, z_d]^T$ and $z_i \in \{0, 1\}$, $i = 1, \dots, d$. Then the Fisher score in (1) can be equivalently reformulated as follows,

$$\begin{aligned} \max_{\mathbf{z}} \quad & \text{tr}[(\text{diag}(\mathbf{z}) X X^T \text{diag}(\mathbf{z}) + \gamma I_{d \times d})^{-1} \text{diag}(\mathbf{z}) X Y Y^T X^T \text{diag}(\mathbf{z})] \\ \text{s.t.} \quad & \mathbf{z} \in \{0, 1\}^d, \mathbf{z}^T \mathbf{1}_d = m. \end{aligned} \quad (3)$$

where $\text{diag}(\mathbf{z})$ is a diagonal matrix whose diagonal elements are z_i . The objective function in (3) can be further simplified as follows:

$$\begin{aligned} \mathcal{L}(\mathbf{z}) &= \text{tr}[Y^T X^T \text{diag}(\mathbf{z}) (\text{diag}(\mathbf{z}) X X^T \text{diag}(\mathbf{z}) + \gamma I_{d \times d})^{-1} \text{diag}(\mathbf{z}) X Y] \\ &= \text{tr}[Y^T X^T \text{diag}(\mathbf{z}) X (X^T \text{diag}(\mathbf{z}) X + \gamma I_{n \times n})^{-1} Y] \\ &= \text{tr}[Y^T (I_{n \times n} - \gamma (X^T \text{diag}(\mathbf{z}) X + \gamma I_{n \times n})^{-1}) Y] \end{aligned} \quad (4)$$

It is easy to verify that $Y^T Y = I$. Therefore, it can be further simplified as

$$\min_{\mathbf{z}} \text{tr}[Y^T (X^T \text{diag}(\mathbf{z}) X + \gamma I_{n \times n})^{-1} Y] \quad \text{s.t.} \quad \mathbf{z} \in \{0, 1\}^d, \mathbf{z}^T \mathbf{1}_d = m. \quad (5)$$

3.2 Spectral Clustering

Spectral clustering is one of the most popular clustering methods in recent years, which exploits the eigen-structure of a specially constructed matrix. Generally, the objective function of spectral clustering [15] algorithms can be defined as:

$$\min_Y \text{tr}(Y^T L Y) \quad \text{s.t.} \quad Y^T Y = I \quad (6)$$

where L is a Laplacian matrix to approximate the manifold structure with different choices. In order to capture the local data structure, one can construct a nearest neighbor graph with an affinity matrix $A \in \mathcal{R}^{n \times n}$. The simplest definition of A is as follows:

$$A_{ij} = \begin{cases} 1, & \mathbf{x}_i \in \mathcal{N}(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i) \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

The normalized Laplacian matrix L is then defined by $L = I - D^{-1/2}AD^{-1/2}$, where D is a diagonal matrix with the diagonal elements as $D_{ii} = \sum_j^n A_{ij}, \forall i$. The optimization problem in (6) can be solved by the eigenvalue decomposition. Based on the continuous solution, the final discrete solution is then obtained by K-means or spectral rotation [15].

4 Proposed Formulation

In this section, we will integrate Fisher score and spectral clustering in a unified framework. The key idea of our method is to find a subset of features, based on which we jointly maximize the Fisher criterion for feature selection and minimize the spectral clustering criterion to best preserve the manifold structure. It can be mathematically formulated as follows,

$$\begin{aligned} & \min_{\mathbf{Y}, \mathbf{z}} \text{tr}(Y^T(L + \lambda(X^T \text{diag}(\mathbf{z})X + \gamma I)^{-1})Y) \\ & \text{s.t. } Y^T Y = I, \mathbf{z} \in \{0, 1\}^d, \mathbf{z}^T \mathbf{1}_d = m, \end{aligned} \quad (8)$$

where λ and γ are two tradeoff parameters. Due to the integer constraints on \mathbf{z} problem (8) is a mixed integer programming [16].

4.1 Optimization of JCFS Algorithm

In this section, we propose to develop an iterative algorithm to solve the optimization problem in (8).

The Computation of \mathbf{Y} for Given \mathbf{z}

$$\min_{\mathbf{Y}} \text{tr}(Y^T(L + \lambda(X^T \text{diag}(\mathbf{z})X + \gamma I)^{-1})Y) \quad \text{s.t. } Y^T Y = I. \quad (9)$$

Though a matrix inverse is involved, we will show that the inverse can be efficiently computed without explicit inverse operation. The above optimization problem integrates Laplacian matrix and inverse covariance matrix based on selected features in a unified framework, which can be solved by the eigenvalue decomposition.

The Computation of \mathbf{z} for Given \mathbf{Y} . Given an existing \mathbf{Y} , the above optimizing with respect to \mathbf{z} is equivalent to solving the following minimization problem

$$\min_{\mathbf{z}} \text{tr}(Y^T(X^T \text{diag}(\mathbf{z})X + \gamma I)^{-1}Y) \quad \text{s.t. } \mathbf{z} \in \{0, 1\}^d, \mathbf{z}^T \mathbf{1}_d = m. \quad (10)$$

The above problem is NP-hard due to the combination nature. To solve it efficiently, we can relax the integer constraints on \mathbf{z} using nonnegative ℓ_1 -norm

regularization. The following relaxed problem is a convex optimization problem which can be solved efficiently.

$$\min_{\mathbf{z}} \quad \text{tr}(Y^T (\sum_{i=1}^d z_i \mathbf{x}^i (\mathbf{x}^i)^T + \gamma I)^{-1} Y) + \mu \mathbf{z}^T \mathbf{1}_d \quad \text{s.t.} \quad \mathbf{z} \geq 0 \quad (11)$$

where $\mu > 0$ is a regularization parameter to control the sparseness of \mathbf{z} . Note that (11) is no longer equivalent to (10) due to the relaxation. In other words, the relaxation can be seen as a tradeoff between the strict equivalence and computational tractability.

Remember that our goal is to selected m features, we did not solve the above convex optimization problem in this paper. Instead, we solve a sequential optimization problem by specifying the number of selected features m , which is particularly convenient for feature selection. Suppose the first t features have been selected, i.e., $\mathbf{x}^1, \dots, \mathbf{x}^t$, then the $(t+1)$ -th feature can be obtained by solving the following problem:

$$\mathbf{x}^{t+1} = \underset{\mathbf{x}^j}{\operatorname{argmin}} \quad \text{tr}(Y^T (\sum_{i=1}^t \mathbf{x}^i (\mathbf{x}^i)^T + \gamma I + \mathbf{x}^j (\mathbf{x}^j)^T)^{-1} Y) \quad (12)$$

By using the Woodbury-Morrison formula [17], we have

$$(\sum_{i=1}^t \mathbf{x}^i (\mathbf{x}^i)^T + \gamma I + \mathbf{x}^j (\mathbf{x}^j)^T)^{-1} = M_t - \frac{M_t \mathbf{x}^j (\mathbf{x}^j)^T M_t}{1 + (\mathbf{x}^j)^T M_t \mathbf{x}^j}. \quad (13)$$

where $M_t = (\sum_{i=1}^t \mathbf{x}^i (\mathbf{x}^i)^T + \gamma I)^{-1}$. Eq. (13) can be used to compute the inverse of M_{t+1} efficiently without inverse operation. Plug (13) into (12), it is equivalent to solving the following problem

$$\mathbf{x}^{t+1} = \underset{\mathbf{x}^j}{\operatorname{argmax}} \quad \frac{\mathbf{x}^j M_t Y Y^T M_t \mathbf{x}^j}{1 + (\mathbf{x}^j)^T M_t \mathbf{x}^j} \quad (14)$$

which is computed on each feature independently. The above process is repeated until we have selected m features. More details about the sequential optimization scheme can be found at [14, 18]. We summarize the complete JCFS algorithm for feature selection in Algorithm (1).

5 Connection to Prior Work

In this section, we discuss the connection between JCFS and SEC, Discriminative K-means, JELSR [9] and DCS [14]. The first two are proposed for clustering task, while the last two are designed for unsupervised feature selection.

5.1 Connection between JCFS and Spectral Clustering

JCFS reduces to spectral clustering, if λ is set as zero. Therefore spectral clustering is a special case of JCFS.

Algorithm 1. JCFS for Feature Selection

Input: data set X , The number of clusters c , The number of selected features m , The number of nearest neighbors k , parameters λ and γ

Output: m selected features.

- 1: Construct the Laplacian matrix L on a nearest neighborhood graph;
- 2: **repeat**
- 3: Solve (9) by the eigenvalue decomposition and obtain the optimal Y ;
- 4: Set $t = 0$ and initialize $M_0 = \frac{1}{\gamma}I$;
- 5: **for** $t = 1$ to m **do**
- 6: Select t -th feature according to (14);
- 7: Compute M_t according to (13);
- 8: **end for**
- 9: **until** Selected features in consecutive iterations are not change

5.2 Connection between JCFS and Discriminative K-Means

Several recent work [11, 12] incorporate supervised dimension reduction such as Linear Discriminant Analysis (LDA) [19] into the clustering framework, which jointly learn the low-dimensional subspace and clustering. For instance, Discriminative Clustering methods solve the following optimization problem:

$$\max_{W,Y} \text{tr}((W^T(S_t + \gamma I)^{-1}W)(W^T S_b W)) \quad (15)$$

There are two sets of variables, the projection matrix W and the scaled cluster assignment matrix Y , in (15). The above optimization problem can be simplified by optimizing Y only based on the following observation [12]:

$$\text{tr}((W^T(S_t + \gamma I)^{-1}W)(W^T S_b W)) \leq \text{tr}((S_t + \gamma I)^{-1}S_b) \quad (16)$$

where the equality holds when $W = VS$, and V is composed of the eigenvectors of $(S_t + \gamma I)^{-1}S_b$ corresponding to all the nonzero eigenvalues, S is an arbitrary nonsingular matrix. Based on (16), JCFS reduces to Discriminant K-means, if $\lambda \rightarrow \infty$ and $m = d$. Therefore Discriminant K-means is a special case of JCFS.

5.3 Connection between JCFS and JELSR

JELSR [9] integrate embedding learning and sparse regression in a unified framework to perform unsupervised feature selection. It can be formulated as follows.

$$\min_{W,Y} \text{tr}(Y^T LY) + \beta(\|X^T W - Y\|_2^2 + \alpha\|W\|_{2,1}) \quad \text{s.t.} \quad Y^T Y = I \quad (17)$$

The sparse subspace W and embedding Y in (17) are learned iteratively. To analysis the connection between JCFS and JELSR, we follow the updating rules of JELSR. Suppose the t -th iteration of W is given by W_t , the $(t+1)$ -th iteration can be obtained by $W_{t+1} = (XX^T + \alpha U)^{-1}XY$, where U is a diagonal matrix and $U_{ii} = \frac{1}{\|2(W_t)_i\|_2}$. Substitute W_{t+1} into (17), the optimization of Y leads to

$$\min_Y \text{tr}(Y^T(L + \beta I_{n \times n} - \beta X^T(XX^T + \alpha U)^{-1}X)Y) \quad \text{s.t.} \quad Y^T Y = I \quad (18)$$

which can be further reformulated into

$$\min_Y \text{tr}(Y^T(L + \beta\alpha(X^T \text{diag}(z)X + \alpha I)^{-1})Y) \quad \text{s.t.} \quad Y^T Y = I. \quad (19)$$

where $z_i = \|\mathbf{w}_i\|_2$. When Y is fixed the optimization of W is solved by the following regularized problem,

$$\min_W \|X^T W - Y\|_2^2 + \alpha \|W\|_{2,1} \quad (20)$$

It is mathematically similar to Group Lasso problem [2] and multi-task feature selection [20]. Based on above formulation, we found that JELSR and JCFS solve similar problem to update Y , the difference lies JCFS only use selected features (0-1 weights of features) while JELSR use all features weighted by its norm $\|\mathbf{w}_i\|_2$. For feature selection, JELSR solve a $\ell_{2,1}$ -norm minimization problem which leads to sparsity of W , however JCFS directly select features to maximizes Fisher score.

Besides, the differences between JCFS and MRSF [21], MCFS [6] can also be obtained based on the above analysis. MRSF and MCFS are two-stage approaches, which both first learn the graph embedding. MRSF then selects features by solving the optimization problem (20), while MCFS selects features via solving K independently LASSO problems.

5.4 Connection between JCFS and Discriminative Codeword Selection

DCS [14] was proposed to minimize the fitting error based on selected features, which can be formulated as follow.

$$\min_{W,b,z} \|Y - X^T \text{diag}(z)W - \mathbf{1}_m b^T\|_F^2 + \alpha \|W\|_F^2 \quad (21)$$

After substituting the optimal value of W and b into (21), the above optimization problem is equivalent to

$$\min_{Y,z} \text{tr}(Y' H_n (\sum_{i=1}^d z_i \mathbf{x}^i (\mathbf{x}^i)^T + \alpha I)^{-1} H_n Y) \quad (22)$$

$$\text{s.t.} \quad Y \in \{0, 1\}^{n \times r}, Y \mathbf{1}_r = \mathbf{1}_n, z \in \{0, 1\}^n, \mathbf{1}_n^T z = m, \quad (23)$$

where $H_n = I_{n \times n} - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T$ is the centering matrix.

It is clear that the above formulation is essentially the same as the discriminative feature selection part in JCFS, however we derive JCFS from supervised feature selection (*Fisher Score*) criterion, while DCS is derived from *ridge regression*. Besides JCFS also integrate spectral clustering to capture manifold information which have been shown to be useful for unsupervised feature selection [4, 6].

6 Experiments

In this section, several experiments are carried out to show the effectiveness of our proposed JCFS for feature selection. We perform clustering and nearest neighbor classification experiments by only using the selected features. The following five unsupervised feature selection algorithms are compared:

- Max Variance which selects those features of maximum variances in order to obtain the best expressive power.
- Laplacian Score [4] which selects the features most consistent with the Gaussian Laplacian matrix.
- Multi-Cluster Feature Selection (MCFS)¹ [6] which selects features using spectral regression with ℓ_1 -norm regularization.
- JELSR [9] which performs feature selection via unifying the graph embedding and sparse regression.
- Our proposed JCFS algorithm.

6.1 Data Sets

We use four real world data sets in our experiments, the processed version of these data sets can be obtained from Cai's page².

The first one is the ORL face database which consists of a total of 400 face images, of a total of 40 subjects (10 samples per subject). The size of each cropped image is 32×32 pixels, with 256 gray levels per pixel. Thus, each face image can be represented by a 1,024-dimensional vector.

The second one is the COIL image library from Columbia. It contains 20 objects. The images of each objects were taken 5 degrees apart as the object is rotated on a turntable and each objects has 72 images. The size of each image is 32×32 pixels, with 256 gray levels per pixel. Thus, each image is represented by a 1,024-dimensional vector.

The third one is Isolet spoken letter recognition data³. This data set was generated as follows. 150 subjects spoke the name of each letter of the alphabet twice. The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1 through isolet5. In our experiments, we use isolet5 which consists 1559 examples with 617 features.

The fourth one is Extended Yale-B database⁴ contains 16128 face images of 38 human subjects under 9 pose and 64 illumination conditions. In our experiment, we choose the frontal pose and use all the images under different illumination, thus we get 2414 image in total. They are resized to 32×32 pixels, with 256 gray levels per pixel. Thus each face image is represented as a 1024-dimensional vector.

¹ The code is available at <http://www.zjucadcg.cn/dengcai/MCFS/index.html>. The code for Laplacian Score can also be found in this page.

² <http://www.zjucadcg.cn/dengcai/Data/data.html>

³ <http://archive.ics.uci.edu/ml/datasets/ISOLET>

⁴ <http://vision.ucsd.edu/leekc/ExtYaleDatabase/ExtYaleB.html>

6.2 Clustering

We perform K-means clustering by using the selected features and compare the results of different algorithms in this test.

Evaluation Metrics. The performance is evaluated by comparing the labels obtained using clustering algorithms with the ground truth labels. We use Clustering Accuracy (ACC) and Normalized Mutual Information (NMI) to measure the clustering performance.

Parameter Settings. Several parameters need to be set beforehand for these algorithms. All the compared algorithms except Max Variance are need to determine the parameter k which specifies the number of nearest neighbors. Generally speaking, k should be set as a small number to preserve the local manifold structure. To fairly compare their performances, we fix k as 5, used in [6, 7], for all the algorithms on all the data sets. For Lap Socre, MCFS, and JCFS we use binary weights to represent the nearest graph for its simplicity. The dimensionality of graph embedding in MCFS and JELSR is fixed as the number of clusters. For JELSR, we fix $\alpha = 1$ and search regularization parameter β from the $\{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1\}$. For our JCFS, we simply set $\gamma = 10^{-4}$ and search the best parameter of λ from the grid $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$. The total number of clusters c is provided for all the clustering algorithms. To reduce the statistical variation of K-means results, we independently repeat the experiments for 100 times with random initializations. We report the best average results corresponding to the best objective values in terms of ACC and NMI respectively.

Clustering Results. Table (1, 2, 3, and 4) show the clustering performance versus the number of selected features m on different data sets. As can be seen, JCFS performs better than the other methods in most cases. Comparing with second best method, JCFS achieves 8.1% (4.2%), 14.1% (10.1%), 4.9% (7.4%), and 8.8% (8.4%) relative improvements in average when measured by ACC (NMI) on the ORL, COIL20, ISOLET, YABLEB data sets, respectively. This show that the idea of integrating supervised feature selection into the clustering framework is beneficial in designing unsupervised feature selection methods. It would be interesting to note that, on these data sets, our proposed algorithm performs surprisingly well by using only very few features, such as 5 and 15 features.

6.3 Nearest Neighbor Classification

In this subsection, we evaluate different feature selection algorithms in the classification task. We perform nearest neighbor classifier (1-NN) with the selected features. Classification accuracy is used to measure the performance. The experimental results are shown in Table 5, we can observe that JCFS is competitive with other algorithms for nearest neighbor classification different selected features m .

Table 1. Clustering performance on the ORL with m selected features

m	ACC					NMI				
	MaxVar	LapScore	MCFS	JELSR	JCFs	MaxVar	LapScore	MCFS	JELSR	JCFs
5	29.4	38.7	36.2	37.2	43.1	54.5	62.2	59.8	60.4	65.9
15	31.1	40.4	47.3	43.3	51.1	56.4	64.3	70.2	66.4	72.8
25	33.2	41.8	50.6	46.7	52.6	58.5	65.4	72.6	69.3	74.1
35	35.1	41.5	49.6	47.8	53.7	60.3	65.6	71.6	70.9	74.9
50	37.2	44.9	51.5	49.2	53.7	61.8	68.6	74.1	71.9	75.0

Table 2. Clustering performance on the COIL with m selected features

m	ACC					NMI				
	MaxVar	LapScore	MCFS	JELSR	JCFs	MaxVar	LapScore	MCFS	JELSR	JCFs
5	33.9	33.4	45.5	43.4	53.0	51.0	50.6	55.3	55.8	63.0
15	40.1	41.1	49.2	53.2	62.5	58.1	59.3	63.3	64.5	73.2
25	45.2	43.4	51.0	53.3	61.2	60.7	62.4	66.2	65.9	74.5
35	46.4	50.1	54.8	54.7	60.5	62.5	65.3	69.7	68.2	74.9
50	47.4	52.8	53.9	56.1	60.2	65.2	67.1	69.6	70.8	74.7

Table 3. Clustering performance on the ISOLET with m selected features

m	ACC					NMI				
	MaxVar	LapScore	MCFS	JELSR	JCFs	MaxVar	LapScore	MCFS	JELSR	JCFs
5	20.3	18.9	26.7	33.1	31.2	29.3	39.5	37.6	50.3	49.4
15	26.4	32.8	46.7	43.4	54.2	38.9	50.6	63.0	59.6	69.6
25	37.2	35.6	52.0	46.8	52.0	51.9	54.2	68.6	63.2	69.6
35	38.0	41.9	51.8	46.5	51.0	54.9	61.8	67.6	65.1	69.4
50	39.7	43.4	53.1	52.1	53.3	59.2	63.4	70.2	68.4	71.9

Table 4. Clustering performance on the YALEB with m selected features

m	ACC					NMI				
	MaxVar	LapScore	MCFS	JELSR	JCFs	MaxVar	LapScore	MCFS	JELSR	JCFs
5	8.9	8.7	15.0	11.0	17.0	13.9	13.8	25.2	17.9	28.0
15	9.2	8.6	14.1	10.5	15.0	15.2	13.6	22.1	18.2	24.0
25	9.2	8.8	14.9	10.3	15.2	13.8	14.1	24.7	17.1	23.3
35	9.2	8.8	13.6	10.2	15.1	13.2	13.9	20.7	16.7	23.3
50	9.1	8.8	13.3	10.1	14.9	13.2	13.8	19.7	16.1	23.4

Table 5. 1-NN classification accuracy on all the data sets with m selected features

DataSet	ORL			COIL			ISOLET			YALEB		
	m	5	25	50	5	25	50	5	25	50	5	25
MaxVar	40.3	62.5	71.3	55.8	78.8	84.5	20.3	57.6	70.5	17.8	31.5	35.7
LapScore	51.2	71.8	83.0	55.8	74.5	83.3	21.0	54.1	73.6	10.4	16.9	20.0
MCFS	49.5	92.0	93.5	80.8	96.4	98.8	26.4	71.6	78.8	30.9	59.5	60.9
JELSR	50.7	85.5	93.3	70.3	96.1	99.2	38.7	71.8	78.3	21.7	44.5	53.0
JCFs	63.2	91.3	94.0	82.7	99.3	99.9	36.1	75.5	80.8	40.3	55.6	64.7

6.4 Parameters Selection

Though our algorithms have three parameters, that are, regularization parameters (λ and γ) and the number of nearest neighbors k . The parameter k is commonly used for manifold learning and spectral clustering, and the results are often stable when k is small. The parameter γ can be safely set as a small number. Due to space limitation, we only show the clustering results with different λ for different selected features m . The data set used for this test is the ORL face database. The experimental results are shown in Fig 1

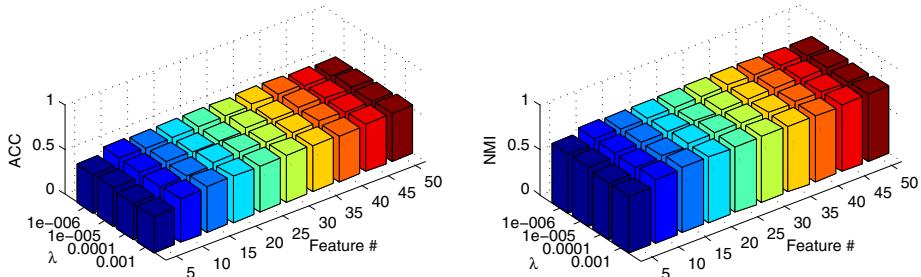


Fig. 1. Clustering performance variation of JCFS on ORL data set w.r.t different λ

7 Conclusions and Future Work

In this paper, we have proposed a new unsupervised feature selection approach which selected features that jointly maximize the supervised Fisher criterion and best preserve the manifold information. We show that spectral clustering, discriminative K-means, DCS are all the special cases of JCFS. We also prove that JELSR and JCFS share similar objective function, where features in JELSR are weighted by its norm while JCFS use a 0-1 weight scheme. Our preliminary results on clustering and classification with selected features show that JCFS outperforms the compared algorithms. In future, we plan to explore other supervised feature selection techniques into the clustering framework for unsupervised feature selection.

Acknowledgments. We would like to thank all anonymous reviewers for their helpful comments. This work is supported in part by NSFC grant 60970045 and China National 973 project 2013CB329305.

References

1. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn., vol. 2 (2001)
2. Nie, F., Huang, H., Cai, X., Ding, C.: Efficient and robust feature selection via joint $\ell_{2,1}$ -norms minimization. In: Advances in Neural Information Processing Systems, NIPS (2010)

3. Nie, F., Xiang, S., Jia, Y., Zhang, C., Yan, S.: Trace ratio criterion for feature selection. In: Proceedings of the 23rd International Conference on Artificial Intelligence (AAAI), vol. 2, pp. 671–676 (2008)
4. He, X., Cai, D., Niyogi, P.: Laplacian score for feature selection. In: Advances in Neural Information Processing Systems (NIPS), vol. 18, pp. 507–514 (2006)
5. Zhao, Z., Liu, H.: Spectral feature selection for supervised and unsupervised learning. In: Proceedings of the 24th ICML, pp. 1151–1157 (2007)
6. Cai, D., Zhang, C., He, X.: Unsupervised feature selection for multi-cluster data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 333–342 (2010)
7. Yang, Y., Shen, H.T., Ma, Z., Huang, Z., Zhou, X.: ℓ_{21} -norm regularized discriminative feature selection for unsupervised learning. In: Proceedings of the 22nd IJCAI, pp. 1589–1594 (2011)
8. Gu, Q., Li, Z., Han, J.: Joint feature selection and subspace learning. In: Proceedings of the 22nd IJCAI, pp. 1294–1299 (2011)
9. Hou, C., Nie, F., Yi, D., Wu, Y.: Feature selection via joint embedding learning and sparse regression. In: Proceedings of the 22nd IJCAI, pp. 1324–1329 (2011)
10. Cai, D., He, X., Han, J.: Spectral regression: A unified approach for sparse subspace learning. In: Seventh IEEE International Conference on Data Mining (ICDM), pp. 73–82 (2007)
11. Ding, C., Li, T.: Adaptive dimension reduction using discriminant analysis and k-means clustering. In: Proceedings of the 24th ICML, pp. 521–528 (2007)
12. Ye, J., Zhao, Z., Wu, M.: Discriminative k-means for clustering. In: Advances in Neural Information Processing Systems (NIPS), vol. 20, pp. 1649–1656 (2007)
13. Bach, F., Harchaoui, Z.: Diffrac: A discriminative and flexible framework for clustering. In: Advances in Neural Information Processing Systems (NIPS), vol. 20, pp. 49–56 (2008)
14. Zhang, L., Chen, C., Bu, J., Chen, Z., Tan, S., He, X.: Discriminative codeword selection for image representation. In: Proceedings of the International Conference on Multimedia, pp. 173–182. ACM (2010)
15. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 22(8), 888–905 (2000)
16. Boyd, S., Vandenberghe, L.: Convex Optimization (2004)
17. Strang, G.: Introduction to Linear Algebra. Wellesley Cambridge Pr. (2003)
18. He, X., Ji, M., Zhang, C., Bao, H.: A variance minimization criterion to feature selection using laplacian regularization. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 33(10), 2013–2025 (2011)
19. Fukunaga, K.: Introduction to Statistical Pattern Recognition, 2nd edn. (1990)
20. Liu, J., Ji, S., Ye, J.: Multi-task feature learning via efficient $\ell_{2,1}$ -norm minimization. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 339–348. AUAI Press (2009)
21. Zhao, Z., Wang, L., Liu, H.: Efficient spectral feature selection with minimum redundancy. In: Proceedings of the 24th AAAI (2010)

A Self-Supervised Framework for Clustering Ensemble

Liang Du^{1,2}, Yi-Dong Shen¹, Zhiyong Shen³, Jianying Wang⁴, and Zhiwu Xu^{1,2}

¹ State Key Laboratory of Computer Science,

Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

² Graduate University of Chinese Academy of Sciences, University of Chinese
Academy of Sciences, Beijing 100049, China

³ Baidu Inc. Beijing 100085, China

⁴ Computing Center of Shanghai University, Shanghai University, Shanghai, China
`{duliang,ydshen,zhiwu}@ios.ac.cn, shenzhiyong@baidu.com,`
`wangjianying@shu.edu.cn`

Abstract. Clustering ensemble refers to combine a number of base clusterings for a particular data set into a consensus clustering solution. In this paper, we propose a novel self-supervised learning framework for clustering ensemble. Specifically, we treat the base clusterings as pseudo class labels and learn classifiers for each of them. By adding priors to the parameters of these classifiers, we capture the relationships between different base clusterings and meanwhile obtain a single consolidated clustering result. In the proposed framework, we are able to incorporate the original data features to improve the performance of clustering ensemble. Another advantage, which distinguishes the proposed framework from the traditional clustering ensemble approaches, is with the generalization capability, i.e. it is able to assign the incoming data instances to the consensus clusters directly based on the original data features. We conduct extensive experiments on multiple real world data sets to show the effectiveness of our method.

Keywords: Cluster Ensemble, Self-Supervised Learning, Logistic Regression.

1 Introduction

Clustering is a common technique for data analysis used in many AI fields, such as machine learning, data mining, pattern recognition. In the last decade, many approaches have been developed to solve the problem of clustering ensemble. Clustering ensemble (See in Figure 1(a)), a.k.a. clustering aggregation [1] or consensus clustering [2], reconciles multiple clustering results ($\lambda_1, \lambda_2, \dots, \lambda_m$) of a data set, coming from multiple base clusterings, into a single consolidated clustering result (λ) using a *consensus function* (Γ). To construct a consensus function, the key challenges include: 1) characterizing the relationships between the base clusterings; 2) conducting a single consensus clustering result.

In the light of these challenges, we propose a novel consensus framework for clustering ensemble, called *Self-Supervised Framework for Clustering Ensemble* (SSCE). The key idea of the proposed framework (See in Figure 1(b)) is to treat X , a matrix induced from base clusterings $(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_m)$ as a *pseudo* data matrix. On the other hand, each individual base cluster assignments could be viewed as *pseudo* class labels. We call the task of learning a set of linear classifiers (with parameters $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_m$) over X and $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_m$ as *self-supervised* learning, since there is actually no supervised information. Viewing from bayesian perspective, we assume the parameters of each classifier share the same prior distribution (with parameter $\boldsymbol{\theta}$). We then call the linear classifier, who take expectation ($\boldsymbol{\mu}$) of the prior distribution as parameters, the *consensus classifier*. Based on this consensus classifier, we are able to *classify* a data instance to a consensus cluster assignment ($\boldsymbol{\lambda}$).

There is a significant feature in current clustering ensemble approaches, i.e., the consensus clustering is directly learned from the base clusterings *without* accessing the original data (\mathbf{X}), which is widely recognized as an advantage of clustering ensemble methods. It is unknown, however, whether we could boost the performance of consensus clustering *with* accessing the original data. Suppose there are incoming data instances, this is another case in which we may also consider the original feature of data instances. It is hard to assign them to the base clusters without accessing the original feature. Therefore, as to our best knowledge, there is no clustering ensemble approach which is capable of handling incoming data instances. This raises another question whether we could assign the incoming data instances to the consensus clusters directly based on their original features.

The proposed SSCE framework could be extended to address the above issues by simply replacing the pseudo data matrix with the original data matrix (See in Figure 1(c)). By this means, SSCE incorporates the original data via characterizing the relationship between the original data and base clusterings. The consensus classifier learned from \mathbf{X} and $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_m$ could be applied to an incoming data instance to obtain its consensus cluster assignment directly.

The main contributions of this paper include:

1. We propose a novel framework to characterize the linear relationship between the base cluster assignments for clustering ensemble.
2. We extend the proposed framework to incorporate the original data for the purpose of increasing the clustering effectiveness.
3. As to our best knowledge, this work is the first to handle the incoming data instances in clustering ensemble.
4. We conduct extensive empirical evaluations with real life data sets to demonstrate the effectiveness of the proposed framework.

2 Related Work

In this section, we introduce some existing works in the fields of clustering ensemble and multi-task learning, which are closely related to the problem studied in this paper.

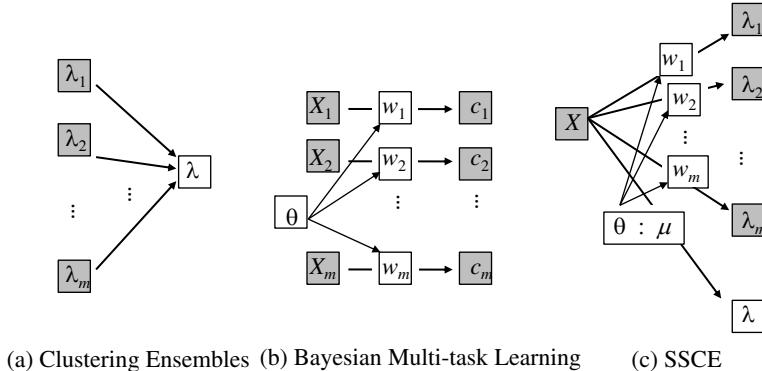


Fig. 1. Shaded and unshaded nodes indicate inputs and outputs, respectively

Clustering Ensemble aims to generate a stable and robust consensus clustering by combining multiple base clustering of a dataset. In general, previous works in this area can be grouped into three categories. The first category is graph based approaches. For instance, [3] proposed three graph-based methods. The cluster-based similarity partitioning algorithm (CSPA) uses METIS to partition the induced similarity graph (vertex = objects, edge weight = cluster-based similarity). The hyper graph partitioning algorithm (HGPA) uses HMETIS to partition the hypergraph (vertex = objects, hyperedge = cluster). The meta clustering algorithm (MCLA) collapses related hyperedges and assigns each object to the collapsed hyperedge in which it participates most strongly. In addition, [4] proposed the hybrid bipartite graph partition algorithm, which partitions the bipartite graph (vertex = objects and cluster) by spectral graph partition. [5] proposed an approach to partition weighted similarity graph.

In the second category the algorithms take advantage of probabilistic graphical models. [6] represents objects as a set of attributes from multiple clusterings, and offers a probabilistic model of consensus using a finite mixture of multinomial distribution in a space of clusterings. [7] proposed bayesian cluster ensemble (BCE), which is a generative probabilistic model for learning cluster ensemble.

The third category is matrix factorization based methods. It has been shown [8] that consensus clustering can be formulated within the framework of nonnegative matrix factorization(NMF). [9] proposed weighted consensus clustering, where each input clustering is weighted in such a way that the final consensus clustering provides a better quality solution. [10] proposed weighted graph regularized NMF method which incorporates both the feature based representation and multiple binary relationships based representation.

Multi-task Learning [11, 12] aims to perform multiple learning tasks together to improve individual performance. Rather differently, in clustering ensemble, we aim to produce a *single* high quality consensus clustering. Moreover, multiple tasks are often performed on different data sets. While, clustering ensemble operated on an *identical* data set to reach a consensus.

3 SSCE

In this section, we propose a novel **S**elf-**S**upervised learning framework for **C**lustering **E**nsemble (SSCE) to produce high quality consensus clusterings.

3.1 Notations and Preliminaries

Suppose we are given a data set with n samples $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ and m base clusterings (or partitions) $\Lambda = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \dots, \boldsymbol{\lambda}_m\}$ of the data. $\boldsymbol{\lambda}_i(\mathbf{x}_j) \in \{1, \dots, k\}$ denote the cluster of \mathbf{x}_j given by the i -th base clustering. We use k to denote the cluster numbers.

In this paper we investigate the usefulness of the features of the data set for clustering ensemble. The data set can be represented by original features, if it is available, or pseudo features. The pseudo data matrix based on base clusterings can be constructed as follows: For each base clustering $\boldsymbol{\lambda}_i \in \mathcal{R}^n$, we construct the binary indicator matrix Λ^i , with a column for each cluster. Entries of this matrix $\Lambda_{j,k}^i = 1$ if the j object is assigned to cluster k . Then we concatenate all the block matrix $\Lambda = (\Lambda^1, \dots, \Lambda^m)$. The pseudo matrix is actually cluster based representation, which is also used in [3] to construct the graph.

3.2 Consistent Labeling

To employ supervised learning approach for clustering ensemble we should align the inconsistency input cluster labels from different base clusterings. In order to achieve the most consistent labeling of clusters between two base clusterings $\boldsymbol{\lambda}_i$ and $\boldsymbol{\lambda}_j$, we must solve an assignment problem, which is also equivalent to a maximum weight bipartite matching problem and can be formulated as follows:

$$\begin{aligned} I_{k \times k} &= \arg \max_I \sum_{i=1}^k \sum_{j=1}^k S_{i,j} I_{i,j} \\ s.t. \sum_{j=1}^k I_{i,j} &= \sum_{i=1}^k I_{i,j} = 1, I_{i,j} \in \{0, 1\} \end{aligned} \quad (1)$$

where $\{S_{i,j}\}$ are the cardinality of intersection of objects labeled i by $\boldsymbol{\lambda}_i$ and objects labeled j by $\boldsymbol{\lambda}_j$, and $\{I_{i,j}\}$ are indicators which determine the correspondence between the clusters in the two partitions. An optimal solution of the problem (1) can be found by Hungarian algorithm [13].

A consistent re-labeling of all the base clusterings can be obtained by using a single reference partition $\boldsymbol{\lambda}_r$. Ideally, the true label is the best choice for a reference $\boldsymbol{\lambda}_r$, however, it is unavailable for clustering ensemble. In practice, any base clustering can be choose as a reference. Then all the remaining base clusterings can be relabeled by solving the problem in Eq. (1) for every pari of partitions $\boldsymbol{\lambda}_r, \boldsymbol{\lambda}_i, i = 1, \dots, l, i \neq r$. Once all the base clusterings are relabeled and aligned, they can be seen as a set of *self supervised* labels of the data set.

3.3 Probabilistic Framework

Given an object \mathbf{x}_j with original or pseudo features and self supervised label L_j^i under i base clustering. we want to find a consensus mapping function $f : \mathcal{X} \rightarrow \boldsymbol{\lambda}$. In this paper we use logistic regression as discriminative model. For 2-class problem the classification model for \mathbf{x}_j under i base clustering can be written in the form

$$P(L_j^i = \pm 1 | \mathbf{x}_j, \mathbf{w}_i) = \sigma(L_j^i \mathbf{w}_i^T \mathbf{x}_j) = \frac{1}{1 + \exp(-L_j^i \mathbf{w}_i^T \mathbf{x}_j)} \quad (2)$$

For multi-class problem, the discriminative model for \mathbf{x}_j under i base clustering takes the form

$$P(L_j^i = k | \mathbf{x}, \mathbf{w}) = \frac{\exp(\mathbf{w}_{ik}^T \mathbf{x})}{\sum_{k'=1}^K \exp(\mathbf{w}_{ik'}^T \mathbf{x})} \quad (3)$$

where \mathbf{w}_{ik} denote the model parameters of class k under i base clustering. These models $\{\mathbf{w}_i\}_{i=1}^m$ can be seen as instances which are generated from a consensus model $\boldsymbol{\mu}$. We use the assumption that the base clusterings are i.i.d. given. The joint distribution of data and model parameters reads

$$\begin{aligned} P(W|X, \theta) &\propto P(X, \theta|W)P(W|\theta) \\ &= \prod_{i=1}^m P(X, \lambda_i | \mathbf{w}_i)P(\mathbf{w}_i | \boldsymbol{\mu})P(W|\theta) \end{aligned} \quad (4)$$

where $P(\mathbf{w}_i | \boldsymbol{\mu})$ is a gaussian prior on each base clustering independently. To model the relationships among these base clusterings we add matrix normal distribution $P(W|\theta) = \mathcal{MN}(W | \boldsymbol{\mu}\mathbf{1}_m^T, I \otimes \Omega)$ [14], where the covariance matrix I_d captures the relationships between features and the covariance matrix Ω models the relationships among different base clusterings. Then the MAP estimation of W and MLE estimation of $\theta = \{\boldsymbol{\mu}, \Omega\}$ can be obtained by minimizing the following objective function

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^m \left\{ \sum_{j=1}^n l(\mathbf{x}_j, L_j^i, \mathbf{w}_i) + \gamma_1 \|\mathbf{w}_i - \boldsymbol{\mu}\|^2 \right\} \\ &\quad + \gamma_2 \text{trace}((W - \boldsymbol{\mu}\mathbf{1}_m^T)\Omega^{-1}(W - \boldsymbol{\mu}\mathbf{1}_m^T)^T) + d \ln |\Omega| \end{aligned} \quad (5)$$

which is the negative log likelihood of posterior of W . The first three terms in Eq. (5) is convex with respect to both W and θ if we take convex loss function. The last term is concave which make the problem difficult to optimize. The last term $\ln |\Omega|$ is used to penalize the complexity of Ω . To simplify the optimize procedure, we replace the last term with a constraint $\text{trace}(\Omega) \leq 1$ to control the complexity, which has been adopted in [11] [12], the above problem becomes

$$\begin{aligned}
\mathcal{L} = & \sum_{i=1}^m \left\{ \sum_{j=1}^n l(\mathbf{x}_j, L_j^i, \mathbf{w}_i) + \gamma_1 \|\mathbf{w}_i - \boldsymbol{\mu}\|^2 \right\} \\
& + \gamma_2 \text{trace}((W - \boldsymbol{\mu} \mathbf{1}_m^T) \Omega^{-1} (W - \boldsymbol{\mu} \mathbf{1}_m^T)^T) \\
\text{s.t. } & \Omega \succeq 0 \\
& \text{trace}(\Omega) \leq 1
\end{aligned} \tag{6}$$

4 Learning Algorithm

Though the optimization problem in Eq. (6) is convex w.r.t. all the variables jointly, it is not easy to optimize the problem w.r.t. all the variables simultaneously. We solve problem Eq. (6) by alternatively minimizing the Eq. (6) with respect to each variable by fixing the others. This procedure is repeated until it converges.

4.1 Optimize W by Fixing μ and Ω

We keep \mathbf{u} and Ω fixed and minimize over W , that is we solve the problem

$$\begin{aligned}
\min_W & \sum_{i=1}^m \left\{ \sum_{j=1}^n l(\mathbf{x}_j, L_j^i, \mathbf{w}_i) + \gamma_1 \|\mathbf{w}_i - \boldsymbol{\mu}\|^2 \right\} \\
& + \gamma_2 \text{trace}((W - \boldsymbol{\mu} \mathbf{1}_m^T) \Omega^{-1} (W - \boldsymbol{\mu} \mathbf{1}_m^T)^T)
\end{aligned} \tag{7}$$

One straightforward way to learn W is to set the gradient w.r.t. W to 0 and solve the corresponding linear system. Because the above problem is convex w.r.t. W , it is also convex w.r.t. \mathbf{w}_i with all other variables fixed. In this paper we adopt an alternative strategy to perform optimize on W , which is to optimize one column of \mathbf{w}_i at a time with the other column fixed. this alternative strategy will be guaranteed to converge to the optimal solution. For 2-class problem, the negative log likelihood of Eq. (2) is

$$l(\mathbf{x}_j, L_j^i, \mathbf{w}_i) = L_j^i \mathbf{w}_i^T \mathbf{x}_j - \log(1 + \exp(L_j^i \mathbf{w}_i^T \mathbf{x}_j)) \tag{8}$$

Hence, the gradient of the above problem with respect to \mathbf{w}_i is

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} = & \sum_{j=1}^n (\delta(L_j^i \mathbf{w}_i^T \mathbf{x}_j) - 1) L_j^i \mathbf{x}_j + 2\gamma_1 (\mathbf{w}_i - \boldsymbol{\mu}) \\
& + \gamma_2 (W - \boldsymbol{\mu} \mathbf{1}_m^T) \Omega^{-1}(:, i)
\end{aligned} \tag{9}$$

For multi-class problem, the negative log likelihoood of Eq. (3) is:

$$l(\mathbf{x}_j, L_j^i, \mathbf{w}_i) = \mathbf{w}_{ik}^T \mathbf{x}_j - \log \left(\sum_{k'=1}^K \exp(\mathbf{w}_{ik'}^T \mathbf{x}_j) \right)$$

The gradient of w.r.t. \mathbf{w}_i is

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} = & \sum_{j=1}^n (\mathbf{w}_{ik}^T \mathbf{x}_j - \log(\sum_{k'=1}^K \exp(\mathbf{w}_{ik'}^T \mathbf{x}_j))) + 2\gamma_1(\mathbf{w}_i - \boldsymbol{\mu}) \\ & + \gamma_2(W - \boldsymbol{\mu}\mathbf{1}_m^T)\Omega^{-1}(:, i) \end{aligned} \quad (10)$$

4.2 Optimize $\boldsymbol{\mu}$ by Fixing W and Ω

By setting the gradient of 6 w.r.t. $\boldsymbol{\mu}$ be 0, we get the close form solution of $\boldsymbol{\mu}$

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^m \mathbf{w}_i \quad (11)$$

4.3 Optimize Ω by Fixing W and $\boldsymbol{\mu}$

Fixing W and $\boldsymbol{\mu}$, Ω is determined by following problem:

$$\begin{aligned} \min_{\Omega} \quad & \text{trace}((W - \boldsymbol{\mu}\mathbf{1}_m^T)\Omega^{-1}(W - \boldsymbol{\mu}\mathbf{1}_m^T)^T) \\ \text{s.t.} \quad & \Omega \succeq 0 \\ & \text{tr}(\Omega) \leq 1 \end{aligned}$$

The close form solution of above problem is given by

$$\Omega = \frac{((W - \boldsymbol{\mu}\mathbf{1}_m^T)^T(W - \boldsymbol{\mu}\mathbf{1}_m^T))^{\frac{1}{2}}}{\text{tr}(((W - \boldsymbol{\mu}\mathbf{1}_m^T)^T(W - \boldsymbol{\mu}\mathbf{1}_m^T))^{\frac{1}{2}})} \quad (12)$$

where the proof can be found in [11] and [12]. The overall approach, called SSCE, is summarized in Algorithm 1.

5 Experiments

In this section, we empirically evaluate the proposed SSCE framework over multiple benchmark data sets. We begin with a description of these data sets with the evaluation metrics, and then provide the evaluation results of the consensus clustering as well as the generalization capability.

5.1 Dataset Description

We carry out our experiments on totally 13 data sets from UCI machine learning repository. These data sets have been widely used in literatures of clustering ensemble, including [7] and [8]. An overview of these data sets, including numbers of instances, features and classes in each data set, is given in Table 1.

Algorithm 1: SSCE algorithm

Input: data matrix X , base clusterings $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_m\}$, γ_1 and γ_2
Output: Cluster labels for data points, classification model μ
Initialization set $\Omega = \frac{I_{m \times m}}{m}$;
while convergence condition is not true **do**
 for $i = 1, \dots, m$ **do**
 | compute w_i using Eq. (9) or Eq. (10);
 | **end**
 update μ using Eq. (11);
 set Ω by Eq. (12)
 end

5.2 Evaluation Measure

We evaluate the clustering ensemble results by comparing the consensus clustering produced by clustering ensemble algorithms with the provided class labels. Specifically, Accuracy of Clustering (AC) is adopted to measure the performance, which discovers the one-to-one relationship between clusters and classes. Given a point x_i , let p_i and q_i be the clustering result and the ground truth label, respectively. The ACC is defined as follows:

$$\text{ACC} = \frac{1}{n} \sum_{i=1}^n \delta(q_i, \text{map}(p_i)), \quad (13)$$

where n is the total number of samples and $\delta(x, y)$ is the delta function that equals 1 if $x = y$ and equals 0 otherwise, and $\text{map}(\cdot)$ is the permutation mapping function that maps each cluster index to a true class label. The best mapping can be found by using the Kuhn-Munkres algorithm [15]. The greater clustering accuracy means the better clustering performance.

5.3 Compared Methods

To demonstrate how the performance of clustering ensemble can be improved by our method, we compare the proposed approach with the results of running k-means on the original data set or the base clusterings (KC). These are often used as baselines to verify the clustering ensemble approaches often produce more accurate and robust results against single clustering methods. We also compare our method with the bayesian cluster ensemble (BCE) in [7], the NMF-based consensus clustering (NMFC) in [8], the cluster-based similarity partitioning algorithm (CSPA), the hyper graph partitioning algorithm (HGPA), and the meta-clustering algorithm (MCLA) described in [3]. For the last three methods, we use the authors' matlab implementation ClusterPack¹. We test our algorithm on data sets with original features (SSCE). we also report the results by experiments on pseudo data matrix (SSCE-P), which is always available for clustering ensemble.

¹ www.lans.ece.utexas.edu/~strehl/

Table 1. Description of 13 data sets

Data sets	Samples	Dimensions	Classes
balance	625	4	3
bupa	345	6	2
glass	214	9	6
ionosphere	351	34	2
iris	150	4	3
magic04	19020	10	2
pima	768	8	2
protein	116	20	6
wdbc	569	30	2
wine	178	13	3
segment	2310	19	7
soybean	47	35	4
zoo	101	18	7

5.4 Experiments on Consensus Clustering

We use similar settings with BCE [7] to show the effectiveness of our proposed algorithm. For all reported results, there are two steps leading to the final consensus clustering. First, given n objects, we run k -means 2000 times with random initialization and obtain 2000 base clustering results, which are further divided into 100 subsets, with an $n \times 20$ base clustering matrix each. Then we run clustering ensemble algorithm 100 times on these subsets. All the data sets have been preprocessed such that each feature has zero mean and unit standard deviation. To simplify our model, we set $\gamma_1 = \gamma_2$ in all the experiments, and the best parameter is obtained by search on the grid of $\{0.01, 0.1, 1, 10, 100\}$.

Table 2 shows the clustering accuracy on the data sets. It is observed that the advantage of the proposed algorithm is much more clear for clustering ensemble. For example, the average improvement of SSCE on original features over BCE, the second best algorithm, achieves 5.7% on all the data sets, the average improvement of our approach on pseudo data matrix (SSCE-P) over BCE is 3.4%. Besides, the proposed approach perform statistically significantly better than the compared methods on 10/13 data sets at 95% significance level. Its success can be explained by the fact that the features (Original or Pseudo) are complementary to the base clusterings. We notice that SSCE failed to achieve comparable performance on iris data set with original features. The reason may lies in the iris data set only have 3 features, which make it less discriminative in a supervised manner. When iris is trained on pseudo data matrix, which has 3×20 features, our algorithm produce comparable results. We also observed that our algorithm perform similar on multi class data sets, and SSCE on original features perform better than its counterpart SSCE-P for 2-class problem.

Table 2. Experimental Results in Clustering Accuracy. The \times entry for CSPA is due to out of memory (4GB).

Data sets	Approaches								
	Kmeans	MCLA	CSPA	HGPA	KC	BCE	NMFC	SSCE-P	SSCE
balance	0.519	0.498	0.516	0.412	0.522	0.513	0.517	0.530	0.526
bupa	0.554	0.554	0.562	0.508	0.554	0.557	0.554	0.580	0.577
glass	0.517	0.464	0.412	0.375	0.467	0.503	0.440	0.553	0.555
ionosphere	0.712	0.712	0.678	0.584	0.712	0.711	0.712	0.712	0.704
iris	0.825	0.893	0.874	0.602	0.796	0.881	0.773	0.885	0.775
magic04	0.649	0.649	\times	0.500	0.649	0.649	0.648	0.649	0.680
pima	0.660	0.660	0.544	0.503	0.660	0.659	0.660	0.660	0.678
protein	0.525	0.574	0.586	0.569	0.522	0.553	0.570	0.538	0.543
segment	0.527	0.548	0.523	0.464	0.515	0.547	0.543	0.557	0.588
soybean	0.706	0.723	0.697	0.726	0.676	0.702	0.616	0.731	0.770
wdbc	0.854	0.854	0.670	0.515	0.854	0.825	0.854	0.854	0.950
wine	0.673	0.702	0.679	0.563	0.651	0.692	0.702	0.702	0.786
zoo	0.690	0.756	0.578	0.574	0.648	0.673	0.639	0.793	0.807

5.5 Experiments on Generalization Capability

One of the advantages of SSCE over other clustering ensemble methods is that it has an explicit mapping function over original features. Since SSCE has explicit mapping function, we can choose part of the data to learn a mapping function and use this mapping function to map the rest of data points to the clusters. To evaluate the generalization capability of SSCE, we design the following experiments. For each data set, we firstly randomly split the data set into two parts (60% and 40%), with the 60% used to train the model and the 40% used as the hold-out test set. Then we run kmeans 100 times on train set to generate the base clusterings. We then run clustering ensemble algorithms 5 times, each run with 20 kmeans as input base clusterings, we predict the cluster label of the test set. This whole procedure is repeated 10 times and the average accuracy, that is over $5 * 10$ results, are reported.

Most of the clustering ensemble methods can not directly predict the label of unseen test data. To do this, we assign the cluster label of unseen data with the label of its nearest cluster center, which is computed from train set and consensus clustering result. [7] is a generative graphic model, which can be used to infer the posterior distribution of the clusters. To do this, we firstly run k -means on test set to construct cluster-based representation, then we use the learned BCE model to infer the cluster assignment of the test data (BCE-Infer). we also report the results of prediction of BCE by nearest cluster center strategy (BCE-NC).

Table 3 summarizes results of the second series of experiments. Similar to results in the first experiment, our proposed approach SSCE usually outperforms the other approaches. For example, the average improvement of SSCE on original features over BCE, the second best algorithm, achieves 4.2% on all the data sets.

Table 3. Experimental Results in Classification Accuracy

Data sets	Approaches									
	Kmeans	CSPA	HGPA	MCLA	KC	BCE-Infer	BCE-NC	NMFC	SSCE	
balance	0.504	0.469	0.490	0.518	0.487	0.480	0.532	0.509	0.470	
bupa	0.547	0.547	0.556	0.546	0.547	0.557	0.556	0.547	0.590	
glass	0.536	0.515	0.473	0.480	0.498	0.550	0.488	0.507	0.561	
ionosphere	0.717	0.624	0.624	0.624	0.624	0.702	0.693	0.704	0.720	
iris	0.818	0.921	0.918	0.804	0.859	0.898	0.681	0.838	0.777	
magic04	0.649	0.649	0.600	0.576	0.649	0.649	0.649	0.648	0.655	
pima	0.660	0.660	0.592	0.589	0.660	0.659	0.660	0.660	0.690	
protein	0.543	0.554	0.582	0.577	0.549	0.544	0.460	0.547	0.555	
segment	0.529	0.558	0.545	0.517	0.522	0.549	0.382	0.545	0.583	
soybean	0.737	0.770	0.759	0.757	0.751	0.764	0.623	0.676	0.826	
wdbc	0.855	0.856	0.907	0.906	0.856	0.847	0.805	0.856	0.945	
wine	0.675	0.696	0.719	0.700	0.689	0.700	0.625	0.696	0.795	
zoo	0.749	0.734	0.719	0.718	0.729	0.735	0.634	0.721	0.839	

6 Conclusions

In this paper, we design a novel consensus function for clustering ensemble. We treat the base clusterings as pseudo class labels and learn base classifiers for each of them. By adding priors to the parameters of these classifiers, we capture the relationships between different base clusterings and meanwhile obtain a single consolidated clustering result. We provide the algorithms to estimate the parameters of the base classifiers as well as the prior parameters, from which we induce the consensus classifier. With empirical evaluations over multiple benchmark data sets, we show that the proposed consensus function outperforms the traditional ones. We also demonstrate we may improve the performance of clustering ensemble via incorporating the original data features. Moreover, we exam the generalization capability of the proposed framework and show its advantage in handling incoming data instances. One area of future work is to investigate optimizing the label correspondence together with the parameter estimation. We may directly handle inconsistent labeling problem from multiple base clustering with different number of clusters.

Acknowledgments. We would like to thank all anonymous reviewers for their helpful comments. This work is supported in part by NSFC grant 60970045 and China National 973 project 2013CB329305.

References

1. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(1), 4 (2007)
2. Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. Machine Learning 52(1), 91–118 (2003)

3. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research* 3, 583–617 (2003)
4. Fern, X.Z., Brodley, C.E.: Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the Twenty-first International Conference on Machine Learning, p. 36. ACM (2004)
5. Al-Razgan, M., Domeniconi, C.: Weighted clustering ensembles. In: Proceedings of 6th SIAM International Conference on Data Mining, pp. 258–269 (2006)
6. Topchy, A., Jain, A.K., Punch, W.: A mixture model for clustering ensembles. In: Proceedings of 4th SIAM International Conference on Data Mining, pp. 379–390 (2004)
7. Wang, H., Shan, H., Banerjee, A.: Bayesian cluster ensembles. *Statistical Analysis and Data Mining* 4(1), 54–70 (2011)
8. Li, T., Ding, C., Jordan, M.I.: Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In: Proceedings of the 7th IEEE International Conference on Data Mining, pp. 577–582 (2007)
9. Li, T., Ding, C.: Weighted consensus clustering. In: Proceedings of the 8th SIAM International Conference on Data Mining, pp. 798–809 (2008)
10. Du, L., Li, X., Shen, Y.-D.: Cluster ensembles via weighted graph regularized nonnegative matrix factorization. In: Tang, J., King, I., Chen, L., Wang, J. (eds.) ADMA 2011, Part I. LNCS, vol. 7120, pp. 215–228. Springer, Heidelberg (2011)
11. Evgeniou, A.A.T., Pontil, M.: Multi-task feature learning. In: Advances in Neural Information Processing Systems, vol. 19, pp. 41–48 (2007)
12. Zhang, Y., Yeung, D.Y.: A Convex Formulation for Learning Task Relationships in Multi-Task Learning. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, pp. 733–742 (2010)
13. Munkres, J.: Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 32–38 (1957)
14. Gupta, A.K., Nagar, D.K.: Matrix variate distributions, vol. 104. Chapman & Hall/CRC (1999)
15. Lovász, L., Plummer, M.: Matching theory. Elsevier Science Ltd. (1986)
16. Fern, X., Brodley, C.: Random projection for high dimensional data clustering: A cluster ensemble approach. In: Proceedings of the 20th International Conference on Machine Learning, pp. 186–193 (2003)

GCBN: A Hybrid Spatio-temporal Causal Model for Traffic Analysis and Prediction

Chaogui Zhang and Jiangtao Ren

Sun Yat-sen University, Guangzhou 510006, China

daguizhang@gmail.com, issrjt@mail.sysu.edu.cn

Abstract. In this paper we propose a binary Bayesian network to model the speed variations for traffic speed prediction. Comparing to continuous graphical models, firstly, our method reduces the complexity of the model. Secondly, we use Granger causality test to determine the structure and parameters of the Bayesian network. Experiments on large GPS data of vans in the freeway network illustrate the good performance of our model.

Keywords: Spatio-temporal, Granger causality test.

1 Introduction

With the development of Intelligent Transportation Systems(ITS), more and more real-time traffic data such as sensor data or GPS data can be easily collected. It is clearly that these large amounts of real-time traffic data provide a solid foundation for traffic speed predictions. There are various methods for traffic speed predictions such as auto-regression [5], Kalman filtering [7], Bayesian networks [14], etc. As discussed in [14], Bayesian network performs better than traditional methods in traffic prediction problems because it takes advantage of the spatio-temporal causal relations of road segments. However, learning the structure and parameters of the Bayesian network is difficult and time-consuming, which means it is difficult to use Bayesian network in a real-time system. To challenge this problem, we attempt to introduce the Granger causality test in the Bayesian network learning. The Granger causality test [6] is a model for determining whether one event is useful in predicting another, aiming at mining the causal information among events. Comparing to the maximum likelihood method, the Granger causality test contains better information for predictions as its definition implies.

In a word, we present a new hybrid method called GCBN (Granger Causality based Bayesian Network) for traffic speed prediction based on Granger causality tests and Bayesian networks. Firstly, we perform Granger causality tests to discover the spatio-temporal causal relations and magnitudes among road segments of the traffic network. Secondly we transform the causal relations into a binary Bayesian network to predict the probability of the speed variation for a specific segment. Thus, the predicted continuous traffic speed is estimated based on a linear model.

The rest of the paper is organized as follows. In Section 2, we introduce the preliminary concepts and definitions in this paper. Section 3 shows the key point of our algorithm in detail, that is, how to perform traffic speed predictions. The integrated model is shown in section 4. Section 5 presents the experiments and discussions. Section 6 gives our conclusions.

2 Preliminary

2.1 Granger Causality Test

In this paper, we consider the causality is time constrained. That is, C_t causes E_{t+1} if and only if $p(E_{t+1}|\Omega_t) > p(E_{t+1}|\Omega_t \setminus \{C_t\})$, where Ω_t is all information in t related to E in $t+1$, meaning the cause is ahead of the effect. And the magnitude that C_t causes E_{t+1} is $w_t^{CE} = p(E_{t+1}|\Omega_t) - p(E_{t+1}|\Omega_t \setminus \{C_t\})$, $w_t^{CE} > 0$.

Granger causality[6] is first proposed by Clive Granger, the Novel prize winner in the area of econometrics, which is based on the intuition that a cause helps predict its effects in the future. For example, a time series C is said to Granger causes E , if the error of the auto-regressive model for E in terms of past values of related information including C , that is $\sigma^2(E_{t+1}|\Omega_t)$, is statistically significantly smaller than $\sigma^2(E_{t+1}|\Omega_t \setminus \{C_t\})$. And the magnitude of causality $w_t^{CE} \propto F(\sigma^2(E_{t+1}|\Omega_t \setminus \{C_t\}), \sigma^2(E_{t+1}|\Omega_t))$, where F is a user-defined function. Then we have $p(E_{t+1}|\Omega_t) - p(E_{t+1}|\Omega_t \setminus \{C_t\}) \propto F(\sigma^2(E_{t+1}|\Omega_t \setminus \{C_t\}), \sigma^2(E_{t+1}|\Omega_t))$, which indicates the existence of some formal transformations between the probability and Granger causality.

The Granger causality test does not indicate the causal effect is positive or negative correlative directly. That is, the result of the Granger causality test does not tell us an increase(decrease) in a cause in the past increases or decreases its effect in the present [2][4]. But in [2], the author suggests to use the sum of the jointly significant Granger coefficients to estimate the direction of variation or the sign of the causal relation.

Note that the Granger causality test is another attempt to define the causality, it is certainly not meant to be equivalent to the true causality [1][10]. But actually, the Granger causality test was successfully adopted by researchers in economics [2][6], neuroscience [3], climate [11] and Internet [9] to help get better understanding.

2.2 Problem Definition

Definition 1. The Traffic Network: The network TN consists of n' segments s , that is $TN = \{s^1, \dots, s^{n'}\}$ and $s^i \cap s^j = \emptyset (i \neq j)$. Each segment consists two directions d ($d = 1$ and 2). So, $TN = \{s^{1,1}, s^{1,2}, \dots, s^{n',1}, s^{n',2}\}$. The segments can be divided according to intersections or toll stations. For convenience, in the following paper, a segment means one segment with a specific direction, then we rewrite $TN = \{s^1, \dots, s^n\}$, where $n = 2n'$.

Definition 2. Speed Variational Rate: The quotient between $|x_{t+1}^i - x_t^i|$ and VT , denoted as $q(s_{t+1}^i) = \frac{|x_{t+1}^i - x_t^i|}{VT}$, where x_t^i indicates the traffic speed of s^i in the time interval t and VT is the highest allowable speed.

For a dataset of speed-based vectors $\mathbf{X} = \{x^i\}_{i=1}^n$, $x^i = \langle x_1^i, \dots, x_t^i \rangle$ represents a speed-based vector, a time series describing the varying traffic speed with respect to s^i , where t is the current time interval. Let \mathbf{W}_t be the causal matrix, where $w_t^{ji} \geq 0$ is the j^{th} column and i^{th} row element of \mathbf{W}_t , representing the magnitude that s^j causes s^i discovered from \mathbf{X} . Let $\mathbf{G}_t = (\mathbf{V}_t, \mathbf{E}_t)$ be a directed and weighted causal graph based on \mathbf{W}_t , where \mathbf{V}_t is a set of nodes representing n segments and \mathbf{E}_t is a set of edges representing causal relations associated with w_t^{ji} . Our main problem is how to take full advantage of \mathbf{G}_t to construct a binary Bayesian network for predicting the speed variational rate $q(s_{t'}^i)$ more efficient and accurate, and then estimate the traffic speed $x_{t'}^i$ in t' , that is \hat{x}_t^i , where $t' > t$ and $i = 1$ to n .

3 Bayesian Networks Based on Granger Causality Tests

Generally, for predicting the speed variational rate $q(s_{t'}^i)$, traditional Bayesian networks can be applied as well. Since $q(s_{t'}^i)$ is continuous, building a Bayesian network with continuous variables is required. For this purpose, one can assume the nodes are Gaussian and the joint probability distribution is GMM, then adopt the EM algorithm to learn the structure and parameters. However, as discussed above, the traditional Bayesian network approach faces problems such as complexity, inefficiency and unreliability. In this paper, we address these problems and present another method to build a Bayesian network for prediction with lower cost.

Suppose the current time interval is t , the basic form in our model to predict the speed variational rate in $t+1$ is shown in figure 1. The parents of a node can be itself(endogenous) or others(exogenous). In detail, after Gran-ger causality tests given the dataset \mathbf{X} from 1 to t (discussed in Section 4), there are h^i segments found as causes of s^i , which means these h^i segments are parents of s^i according to the Bayesian networks' interpretation. We denote them as ψ^i . Then we have $\psi^i = \{s^1, \dots, s^j, \dots, s^{h^i}\}, |\psi^i| = h^i$.

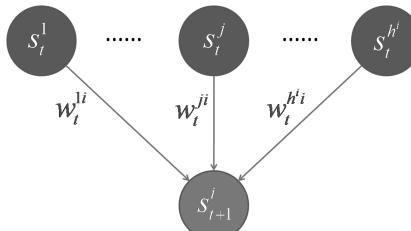


Fig. 1. The basic form for prediction in our model

The feature of our model is discrete. We discretize the continuous speed variational rate $q(s_t^i)$ into binary, that is 0 and 1 denoting the minimum and maximum, respectively. Then, the complexity of the model is greatly reduced. Thus, the observational probabilities and the binary conditional probability table(CPT) are necessary in our model.

Firstly, since the traffic speeds of current and previous time interval are known to us, we can estimate the speed variational rate of each parent node. For the sake of reserving necessary information for computing \hat{x}_{t+1}^i based on $q(s_{t+1}^i)$, we let the continuous value of speed variational rate be the probability of $q(s_t^j) = 1$, that is:

$$p(q(s_t^j) = 1) = \frac{|x_t^j - x_{t-1}^j|}{VT} \quad (1)$$

where VT is the highest allowable speed which is set to 120km/h in data pre-processing. The meaning of (1) is that the higher speed variational rate, the higher probability of reaching the maximum.

Secondly, the CPT is estimated as follow. According to *D-Separation* and the definition of time constrained causality, although s^i in $t + 1$ is not independent of s^j in t , they are independent in the same time interval t . Therefore, s_t^j is independent of each other for $j = 1$ to h^i . Then, we can use a specified function such as linear function or exponent function to model the CPT, by the linear weighted superposition assumption. Due to the limit of space, here we only use conditional multinomial distribution [12](also known as *softmax*) to model the CPT.

$$p(q(s_{t+1}^i) = 1 | \mathbf{c}(\psi^i)_r) = \frac{\exp(\beta_t^i + \mathbf{c}(\psi^i)_r \mathbf{w}_t^{i^T})}{\sum_{r=1}^{2^{h^i}} \exp(\beta_t^i + \mathbf{c}(\psi^i)_r \mathbf{w}_t^{i^T})} \quad (2)$$

where $\mathbf{c}(\psi^i) = \{\mathbf{c}(\psi^i)_1, \dots, \mathbf{c}(\psi^i)_r, \dots, \mathbf{c}(\psi^i)_{2^{h^i}}\}$ is a set of binary row vectors, denoting all kinds of binary combinations of $q(s_t^j)$ for $j = 1$ to h^i orderly. And β_t^i is the offset factor, \mathbf{w}_t^i is the i^{th} row vector of \mathbf{W}_t but only the elements larger than 0 remain. For example, when $h^i = 2$ and set $Z = \sum_{r=1}^{2^{h^i}} \exp(\beta_t^i + \mathbf{c}(\psi^i)_r \mathbf{w}_t^{i^T})$, then the CPT is estimated as table 1 shows. Therefore, according to the Bayesian networks' theory, we can have:

$$p(q(s_{t+1}^i) = 1) = \sum_{r=1}^{2^{h^i}} \frac{\prod_{j=1}^{h^i} p(q(s_t^j)) \exp(\beta_t^i + \mathbf{c}(\psi^i)_r \mathbf{w}_t^{i^T})}{\sum_{r=1}^{2^{h^i}} \exp(\beta_t^i + \mathbf{c}(\psi^i)_r \mathbf{w}_t^{i^T})}. \quad (3)$$

Then, $q(s_{t+1}^i)$ can be calculated as:

$$q(s_{t+1}^i) = 1 * p(q(s_{t+1}^i) = 1) + 0 * (1 - p(q(s_{t+1}^i) = 1)). \quad (4)$$

After obtaining $q(s_{t+1}^i)$, we should infer the direction of variation. According to [2], the direction of variation or the sign of the causal relation can be estimated by the sum of the jointly significant Granger coefficients. Therefore, we can derivate:

$$U = \sum_{j=1}^{h^i} w_t^{ji} (x_t^j - x_{t-k}^j) \sum_{l=1}^k \gamma_l^{ji} \quad (5)$$

Table 1. An example of CPT, where s_t^1 and s_t^2 are the parent nodes of s_{t+1}^1

$q(s_t^1)$	$q(s_t^2)$	$p(q(s_{t+1}^1) = 1)$
0	0	$\frac{\exp(\beta_t^1)}{Z}$
0	1	$\frac{\exp(\beta_t^1 + w_t^{21})}{Z}$
1	0	$\frac{\exp(\beta_t^1 + w_t^{11})}{Z}$
1	1	$\frac{\exp(\beta_t^1 + w_t^{11} + w_t^{21})}{Z}$

where γ_l^{ji} is the Granger coefficient, and k is the maximal lag in the Granger causality test (discussed in Section 4). Then, we can compute the predicted speed \hat{x}_{t+1}^i as:

$$\hat{x}_{t+1}^i = x_t^i + \text{sign}(U)q(s_{t+1}^i)VT \quad (6)$$

where $\text{sign}(U) = 1$ if $U \geq 0$, $\text{sign}(U) = -1$ otherwise.

4 Integration

As we have presented the key point of our method in the above paper, in this section, we would like to illustrate the integrated model. After data pre-processing, causal information generation and Bayesian network construction are the two main tasks.

4.1 Causal Information Generation

Since the speed-based vectors are time series themselves, Granger causality tests are excellent to discover the causal relations among segments and then generate the causal information, including \mathbf{W}_t and \mathbf{G}_t .

Suppose there are h^i segments have direct spatial relations to s^i . That is, if s^j has direct spatial relation to s^i , then it is reachable from s^i to s^j or from s^j to s^i without any other segments in the network. Specially, a segment has direct spatial relation to itself. We denote these h^i segments as ψ^i . Then, for mining the causes of s^i , the Granger causality test is performed by conducting the following regressions:

$$x_t^i = \sum_{j=1}^{h^i} \sum_{l=1}^k \gamma_l^{ji} x_{t-l}^j + \varepsilon_t^i \quad (7)$$

where ε_t^i is an uncorrelated white-noise series and k is called maximal lag, the number of past data points that are used to predict the current data point in t . Ideally k can be equal to infinity, but in practice, k is usually finite and more smaller than the length of given time series. Generally, the best k can be decided by BIC or AIC criterion, or simply by user-setting. Next, the F-test is performed where the null hypothesis is s^j does not Granger causes s^i , that is, $\gamma_l^{ji} = 0$ for

$l = 1$ to k . Then, the p value obtained from F-test can tell us whether or not the null hypothesis is established. If not, then, we can obtain the result that s^j Granger causes s^i given the historical data from 1 to t . Moreover, we can measure the magnitude of Granger causality by the log ratio of the prediction error [13], so:

$$w_t^{ji} = \log \frac{\sigma^2(x^i|\psi^i \setminus \{s^j\})}{\sigma^2(x^i|\psi)}. \quad (8)$$

When $j \neq i$, it means the general Granger causality. When $j = i$, the causality is called Granger autonomy [13], that is, to identify if s^i Granger causes itself. In other words, Granger autonomy means the traffic speed of s^i is affected by its own past besides others' related to it. In particular, if s^j does not Granger causes s^i given the historical data from 1 to t , w_t^{ji} is set to 0.

Then, the causal graph \mathbf{G}_t based on \mathbf{W}_t can be built after Granger causality tests. Figure 2(a) shows an example of Granger causal graphs. Furthermore, a chart of Granger causal flow [8] is helpful to illustrate the causal relations in another way by calculating the result of the weighted out-degree minus weighted in-degree. For example, the causal flow of Node s^4 in figure 2(a) is $3+3-(1+3)=2$. The nodes with positive or negative flow is called sources or sinks. The bigger sources or sinks are worth paying more attention as they are usually the main causes or the main effects of incidents.

4.2 Bayesian Network Construction

Granger causal graphs do not consider the temporal dimension. Therefore, most of time Granger causal graphs are only used to make a further understanding qualitatively as figure 2 shows, but not used to make predictions quantitatively. Since the cause is ahead of its effects, we can assume the delay time is equal to the user-set time interval such as fifteen minutes to one hour. Under this view, we can attach a temporal attribute to each node in the graph, and transform the graph into a Bayesian network for prediction. Figure 2(b) illustrates this idea when we observe the traffic status in t and would like to make predictions in the future. And the model for prediction is already discussed in Section 3.

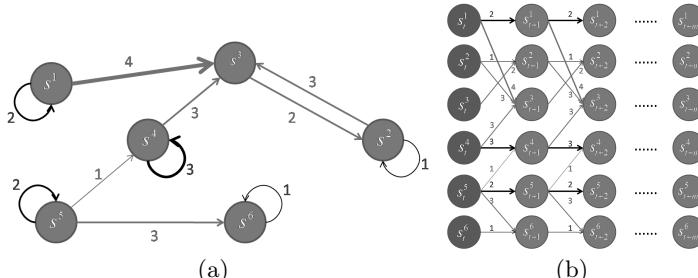


Fig. 2. (a) presents an example of Granger causal graphs with six nodes. The thickness of edges or the numbers indicate the magnitudes of Granger causality or autonomy. (b) The Bayesian network based on the Granger causal graph shown in (a).

4.3 GCBN

Here we summarize our model GCBN, described in Algorithm 1. Since the data pre-processing procedure is not the main point in this paper, we just consider the input as a set of GPS data of size N after data pre-processing procedure collected in t , where t is the current time interval. The time complexity of GCBN can be analyzed as follows: (a) In step 1, we need $O(N)$ operations to analyze the data quantity and generate the speed-based vectors. (b) In step 2-13, the main step is executing the Granger causality tests, where $O(|\psi|*(m-k)*(|\psi|*k)^2)$ operations is needed to execute vector autoregression $|\psi|+1$ times¹. Since $n \geq |\psi|$ and $m \gg k$ in practice, the overall time complexity of GCBN is $MAX(N, n*m*k^2 * |\psi|^3)$. This suggests our method can be applied in different network of diverse scales and used in an online setting when we set m and k appropriately.

Algorithm 1. GCBN

Input:

D : a set of GPS data of size N after data pre-processing collected in t ; AM : adjacency matrix; $round$: the step length for prediction; α : confidence coefficient; β : offset factor; k : maximal lag; m : length of time series.

Output:

pts : the predicted traffic speeds;

- 1: **for** each s^i , $i = 1$ to n **do**
- 2: find ψ^i having direct spatial relations to s^i ;
- 3: run the Granger causality tests by (7) - (8);
- 4: **for** each $t' = t + 1$ to $t + round$ **do**
- 5: obtain the predicted speed variational rates by (4);
- 6: infer the the directions of variation by (5);
- 7: **return** output pts by (6)
- 8: **end for**
- 9: **end for**

5 Experiments

In this section, we report the empirical analysis and prediction evaluation of our model on real traffic GPS data collected by traffic agencies.

5.1 Setting

In this paper, we focus one of the most important regions of the freeway network in Guangdong province, China. The raw traffic GPS data for experiment are generated by 89007 probe vehicles from March 1, 2010 to July 31, 2010, and the data quantity reaches 5113503 totally. We set 1 time interval as 15 minutes, then the average data quantity of a segment in a time interval is 10. We define the maximal length of time series as 432 time intervals(3 days) for Granger causality tests. The maximal lag is selected by BIC. And we set offset factor $\beta = 0$.

¹ Least square method is adopted to solve the vector autoregression problem.

5.2 Causal Analysis

We set the system update frequency as 24 time intervals(6 hours) and run GCBN from March to July. Besides, for better comparison, we also use Bayes Net Toolbox² to learn a traditional Bayesian network(TBN), where Gaussian distributions are used to model continuous-valued nodes. Figure 3. shows the graphs for five months, generated by GCBN and TBN respectively. For TBN, it is frequent that one segment is independent to other segments in our experiment, even itself in previous time, and then the existent probabilities of edges are much lower. As for GCBN, the darker colors of edges indicate that, with Granger causality tests, GCBN can find more causal patterns than TBN when learning the structure every time, which means GCBN can use more advantageous information to predict the traffic speed better.

Figure 4 shows the charts of Granger causal flow for each month and all months in GCBN. We can see that the segments with higher causal weight lie between two main intersections, the closer to the intersection, the higher causal weight as a whole. That means those segments are sources and it is at higher risk when congestion occurs in those segments. On the other hand, the sinks are up streams of sources. Then, the main causal pattern is that segments around the intersection causes their up streams, which is reasonable because the traffic stream is from upstream to downstream and congestion in upstream will block the stream. Besides, the causal pattern is stable for each month, which means the causalities or autonomies are reliable. Therefore according to the above discussion, with the help of charts of Granger causal flow, we can locate the key causes and effects more convenient.

5.3 Prediction Evaluation

Here, we evaluate our method GCBN to predict the traffic speeds comparing with auto regression(AR), Kalman filter(KF) and the traditional Bayesian network(TBN).

For investigating the performance of our method, tests are performed on the following two evaluation metrics:

Root Mean Square Error (RMSE). RMSE [14] is widely used to quantify the difference between the predicted values and the true values, which is defined as:

$$RMSE(\hat{x}_{t+1}, x_{t+1}) = [\frac{1}{n} \sum_{i=1}^n (\hat{x}_{t+1}^i - x_{t+1}^i)^2]^{1/2}. \quad (9)$$

Root Mean Square Error with Cost(RMSEC). In the congestion prediction problem, it is intuitive that the cost is much bigger when we don't predict true congestion than when we predict false congestion. Thus, in the speed prediction problem, the cost is much bigger if any predicted error occurs when the true traffic speed is lower. Therefore, we set up a cost matrix for better evaluating the accuracy of prediction:

² <http://code.google.com/p/bnt/>

$$RMSEC(\hat{x}_{t+1}, x_{t+1}) = \left[\frac{1}{n} \sum_{i=1}^n c_{t+1}^i (\hat{x}_{t+1}^i - x_{t+1}^i)^2 \right]^{1/2} \quad (10)$$

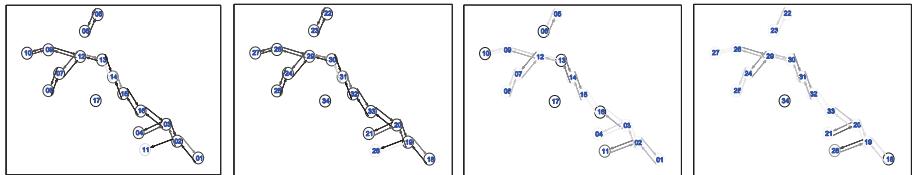
Table 2. The cost matrix used in this paper

		true traffic speed x_{t+1}^i	
		$0 \leq x_{t+1}^i < 20$	$20 \leq x_{t+1}^i < 40$
c_{t+1}^i		5	10

Predicted Result Analysis. In this experiment, we first focus on 3 important holidays in China as rush hours. The result in figure 5 shows that our method GCBN outperforms AR, KF and TBN on all the two evaluation metrics. Especially, on RMSEC, our method GCBN achieves 17.71%, 40.42% and 35.74% relative improvements compared with AR, KF, TBN respectively, which indicates the good performance of GCBN to predict the traffic speed when the probability of congestion is higher. Secondly, we randomly select 30 moments from March to July. GCBN shows better performance as well. As a whole, our method performs better than AR, KF , TBN on RMSE, RMSEC at both ordinary and specific times.

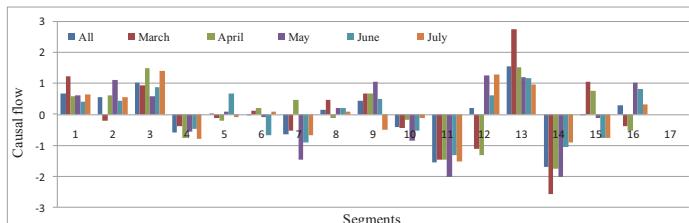
Figure 6(a) shows the predicted results of four methods with respect to s^1 for ten time intervals, and the observed speeds of s^1 and s^3 are also shown in the figure. As we can see, GCBN discovers the spatio-temporal causal relation between s^3 and s^1 , which makes GCBN predict the traffic speed of s^1 in the sixth time interval much more accurate than others. This suggests that our method is more sensitive to the spatio-temporal causal relation among segments than TBN, which brings great benefits to the traffic speed or congestion prediction, especially when the probability of congestion is higher.

Computational Complexity Analysis. Figure 6(b) presents the comparison of computing time for four methods. AR's computing time of learning or predicting is least as it is the most simple method in our experiment. Compared to KF and TBN, our method's computing time is much smaller because our method can learn the structure and parameters at the same time and there is no any iterative process in our method. Figure 6(c) presents the computing time of GCBN for learning or predicting under different number of segments. When the number of segments increases, the computing time of GCBN grows almost linearly. And the analysis of computational complexity in Subsection 4.5 shows our method only need linear time to analyze the data quantity and generate the speed-based vectors, which indicates our method can be applied in different network of diverse scales and used in an online setting.

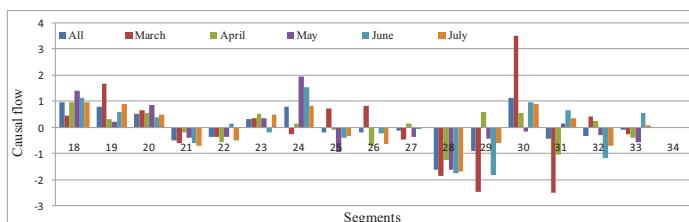


(a) GCBN in Direction1 (b) GCBN in Direction2 (c) TBN in Direction1 (d) TBN in Direction2

Fig. 3. (a) - (d) present the probabilities of occurrence of causalities for five months. The darker colors of edges or circles indicate the higher values.

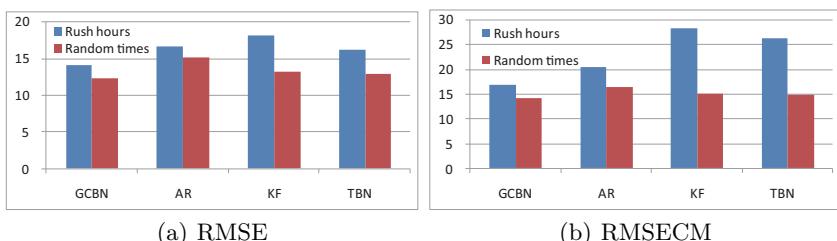


(a) Direction1



(b) Direction2

Fig. 4. The charts of Granger causal flow for each month and all months in GCBN



(a) RMSE

(b) RMSECM

Fig. 5. Accuracy comparison of four method for predicting the traffic speeds in rush hours and random time

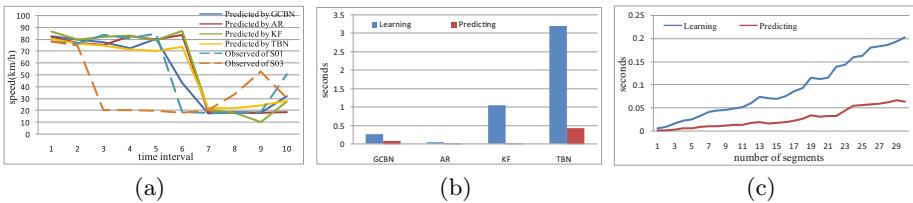


Fig. 6. (a) shows the predicted results of four methods with respect to s^1 for ten time intervals. (b) presents the comparison of computing time for four methods. (c) presents the computing time of GCBN under different number of segments.

6 Conclusions

In this paper, we present a new hybrid method called GCBN for both traffic analysis and prediction based on Granger causality tests and Bayesian networks. Firstly, for reducing the complexity of the model and reflecting the essential causal relations in traffic network, we propose to predict the probability of the speed variation for a segment by a binary Bayesian network. Secondly, we introduce the conception and methodology of Granger causality from the economics field, and adopt the Granger causality test to determine the structure and parameters of the Bayesian network. In comparison with three methods, GCBN demonstrates excellent performance as Granger causality tests can help discover the spatio-temporal causal relations among segments more accurate. Besides, the analysis of computational complexity shows, the operations grows almost linearly when the number of segments increase and our method can be applied in different network of diverse scales and used in an online setting.

Acknowledgements. This work is supported by the Fundamental Research Funds for the Central Universities under Project No. 12lgpy40 and Guangdong Natural Science Foundation under Project No. S2012010010390.

References

- Arnold, A., Liu, Y., Abe, N.: Temporal causal modeling with graphical granger methods. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2007, pp. 66–75 (2007)
- Casu, B., Girardone, C.: Does competition lead to efficiency? The case of eu commercial banks. SSRN eLibrary (2009)
- Ding, M., Chen, Y., Bressler, S.L.: Granger Causality: Basic Theory and Application to Neuroscience, pp. 437–460. Wiley (2006)
- Ge, T., Feng, J., Grabenhorst, F., Rolls, E.T.: Componential granger causality, and its application to identifying the source and mechanisms of the topdown biased activation that controls attention to affective vs sensory processing. NeuroImage (2011)
- Ghosh, B., Basu, B., O’Mahony, M.: Multivariate short-term traffic flow forecasting using time-series analysis. IEEE Transactions on Intelligent Transportation Systems, 246–254 (2009)

6. Granger, C.W.J.: Investigating causal relations by econometric models and cross-spectral methods. *Econometrica* 37(3), 424–438 (1969)
7. Iwao, O., Yorgos J., S.: Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological* 18(1), 1–11 (1984)
8. Seth, A.K.: A matlab toolbox for granger causal connectivity analysis. *Journal of Neuroscience Methods* 186(2), 262–273 (2010)
9. Kim, Y., Balani, R., Zhao, H., Srivastava, M.B.: Granger causality analysis on ip traffic and circuit-level energy monitoring. In: *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, BuildSys 2010*, pp. 43–48 (2010)
10. Liu, Y., Niculescu-Mizil, A., Lozano, A.C., Lu, Y.: Learning temporal causal graphs for relational time-series analysis. In: *Proceedings of the 27th International Conference on Machine Learning, ICML 2010*, pp. 687–694 (2010)
11. Lozano, A.C., Li, H., Niculescu-Mizil, A., Liu, Y., Perlich, C., Hosking, J., Abe, N.: Spatial-temporal causal modeling for climate change attribution. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009*, pp. 587–596 (2009)
12. Murphy, K.P.: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, UC Berkeley, Computer Science Division (July 2002)
13. Seth, A.K.: Measuring autonomy and emergence via granger causality. *Artif. Life* 16, 179–196 (2010)
14. Sun, S., Zhang, C., Yu, G.: A bayesian network approach to traffic flow forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 124–132 (2006)

An Overlapped Community Partition Algorithm Based on Line Graph

Zhenyu Zhang, Zhen Zhang, Wenzhong Yang, and Xiaohong Wu

School of Information Science and Engineering, Xinjiang University, Urumqi, China
xjuzzzy@163.com

Abstract. Overlapped communities detection in complex networks is one of the most intensively investigated problems in recent years. In order to accurately detect the overlapped communities in these networks, an algorithm using edge features, namely SAEC, is proposed. The algorithm transforms topology graph of nodes into line graph of edges and calculates the similarity matrix between nodes, then the edges are clustered using spectral analysis, thus we classify the edges into corresponding communities. According to the attached communities of edges, we cluster the nodes incident with the edges again to find the overlapped nodes among the communities. Experiments on randomly generated and real networks validate the algorithm.

Keywords: community partition, overlapped nodes, spectral analysis, line graph.

1 Introduction

Complex networks constitute an efficacious formalism to represent the relationships among the objects composing many real world systems. One crucial step when studying the structure and dynamics of networks is to identify communities, i.e. the division of a network into groups of nodes having dense interconnections and sparse interconnections. Communities in networks often overlap [1], i.e. some nodes simultaneously belong to several groups. Searching for overlapped communities is usually much more computationally demanding than detecting standard partitions.

In this paper we propose a new algorithm, named SAEC. We calculate the similarity between edges and use spectral analysis to derive overlapped communities partition. Experiments on randomly generated and real networks show the capability of the SAEC approach to correctly detect communities compared to the k-clique approaches.

2 Related Work

Commonly used community detection algorithm[2-6] can be partitioned to two categories: agglomeration method and splitting method. Recently some improvement methods were put forward to detect overlapped communities, e.g. K-Clique

algorithm[1]. These algorithms are designed to reveal overlapped communities in the network, but they also have some limitations, such as parameter selection.

Now there are some methods of communities division employing Spectral analysis[7-9], Many experiments showed that spectral analysis is better than the common clustering algorithm, especially in the high-dimensional data. Spectral analysis commonly used heuristics-based eigen gap to determine k. Ideally, if a data set containing k clusters, there is a big gap between the first k larger eigenvalues and the back k+1 smaller eigenvalues of corresponding laplacian matrix L, be termed eigen gap. We can arrange eigenvalue λ_n from big to small order based on this heuristic rule and find the value of k that satisfies $\max(\lambda_k - \lambda_{k+1})$. Meila and Shi[10] interpreted the similarity between two nodes as random walk probability of Markov chain and tried to use eigenvector of transfer probability matrix $P = D^{-1}W$ to cluster nodes. Azran and Ghahramani[11] present the k value that based on the eigen gap of transfer probability matrix after the M step random walk is more accurate and more close to the real number of clustering than that based on the eigen gap of laplacian matrix L.

3 SAEC Algorithm

SAEC (Spectral Analysis based on Edge Clustering) use line graph[12] to represent this relationship and cluster these edges which reflect this relationship by spectral analysis. Since one node may be associated with multiple edges, as different edges are clustered into different communities, nodes are accordingly partitioned to different communities, so we can find the overlapped nodes by this means.

In order to implement the SAEC algorithm, we need to construct the similarity matrix. We consider the similarity matrix as the set of all similarity between data points. Here we define the similarity between two nodes as formula 1. As shown in the formula 1, if two nodes e_{ik} and e_{jk} which are in line graph share the same node k, this two nodes similarity can be understood as the number of common neighbor nodes of them. in order to avoid the emergence of isolated nodes and achieve relatively average partition, $\min\{n_i, n_j\}$ is added as the denominator in formula 1. Obviously if the two nodes of line graph do not enjoy sharing node, then their similarity is equal to 0. If n_i represents the set of neighbor nodes i and E represents the set of all edges of original node topology graph, then $n_i = |\bigcup_{e_{ij} \in E} j|$.

$$S(e_{ik}, e_{jk}) = \frac{|n_i \cap n_j|}{\min\{n_i, n_j\}} \quad (1)$$

W_{uv} represents similarity of line graph, thus:

$$W_{uv} = S(u, v) \text{ If } u \text{ and } v \text{ have the shared node, } W_{uv} = 0 \text{ otherwise} \quad (2)$$

Transfer probability matrix $P = D^{-1}W$, D is diagonal matrix, the definition as follows:

$$D = \left\{ d_{ii} = \sum_{j=1}^n w_{ij}, d_{ij} = 0 \right\} \quad (3)$$

SAEC performs the following steps:

- (1) Construct the similarity matrix according formula (1).
- (2) Calculate the eigenvalues and eigenvectors $\{\lambda_n, V_n\}_{n=1}^N$ of transfer probability matrix $P = D^{-1}W$ that based on random walk.
- (3) Arrange the eigenvalues from big to small order and find out k values that satisfy $\max_k(\lambda_k - \lambda_{k+1})$.
- (4) Construct an $n*k$ matrix H according to the corresponding eigenvector of the k first eigenvalues, each row represents a node in H, and the number of columns represents the node dimension.
- (5) Call K-means algorithm for matrix H and partition node i to corresponding community, if and only if the line which i corresponds to was assigned to community C_n .
- (6) Pick out the overlapped nodes whose community property are greater than 1 and indicate that which communities these overlapped nodes belong to.

4 Experimental Results

In order to validate SAEC, we employ the network data sets that randomly generated by the computer and the real network data sets.

Fig.1 is the description of the network diagram randomly generated by computer. Fig.1(a) is the experimental results by using K-Clique algorithm. Fig.1(b) is the experimental result by using SAEC algorithm that proposed in this paper. Different colors represent different communities, red triangle nodes denote overlapped nodes.

As can be seen from the graph, K-Clique algorithm can detect the overlapped community structure, the overlapping nodes are {3,10,11,12}. The SAEC algorithm based on the transfer probability matrix P reaches the maximal eigenvalues gap $\max\{\lambda_k - \lambda_{k+1}\} = 0.2248$, $k=4$, so we partition to four communities. In addition to the overlapped nodes {3,10,11,12}, the other overlapped nodes {4,17,22} are detected by SAEC algorithm, this is due to SAEC algorithm excavate communities based on

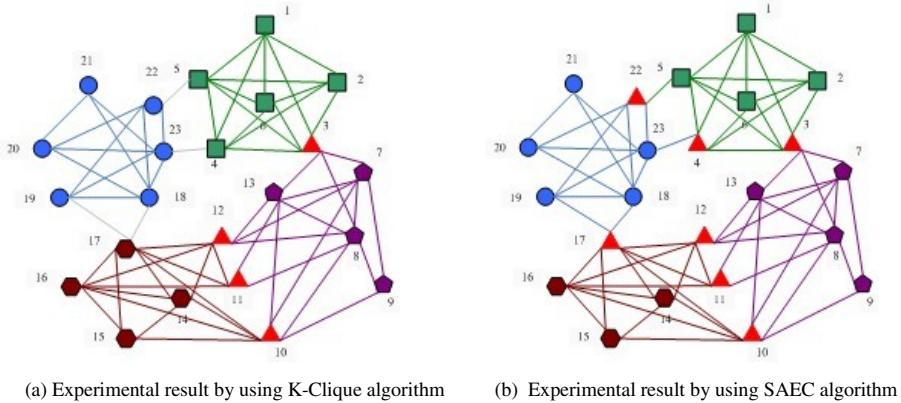


Fig. 1. Communities partition results for randomly generated network

the edge information. The edges like edge (5,22) is partitioned to the communities, so the nodes in edge's two ends are also partitioned to a community. SAEC algorithm partition edge (5,22) to green lines indicated community, meanwhile edge (20, 22) is partitioned to blue lines indicated community, so there is no doubt that the node 22 is a overlapped node.

In real data sets, we use part of data in scientific collaboration network that provided by CFinder¹. As shown in Fig.2, different colors represent different communities, and red triangle nodes represent overlapped nodes. Using SAEC algorithm and K-Clique algorithm separately, the results show that communities structure and overlapped node for scientific collaboration network are the same too.

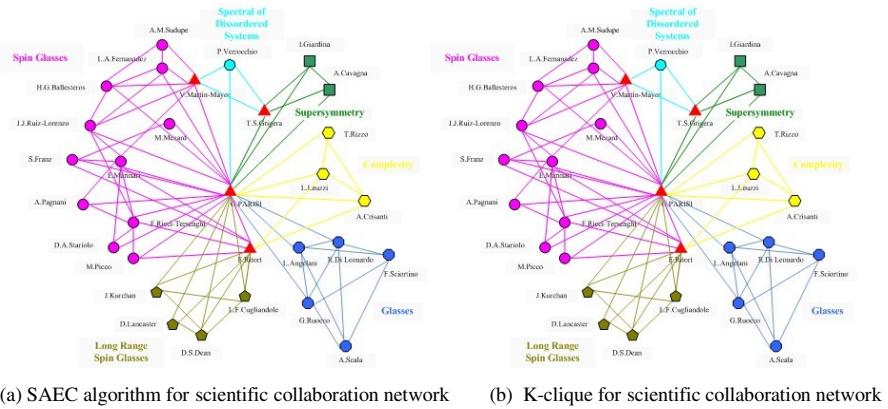


Fig. 2. Communities partition results for scientific collaboration network

¹ CFinder is a free software for finding overlapped dense groups of nodes in networks, based on the Clique Percolation Method.

5 Conclusions and Discussions

The paper presented an algorithm, namely SAEC, to detecting overlapped communities in complex networks. Experiments on randomly generated and the scientific collaboration network showed the capability of the SAEC algorithm to correctly detect overlapped communities. Future research will aim at applying other factors, e.g. the node preference, and deriving more accurate and effective communities division.

Acknowledgments. This work was funded by the National Natural Science Foundation of China under grant number 61262089 and 61262087, Key Project of Xinjiang College Teachers Research(Project No. XJEDU2012I09) and Xinjiang University PhD Scientific Research Fund(Project No. BS110127).

References

1. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 814–818 (2005)
2. Newman, M.E.J.: Communities, modules and large-scale structure in networks. *Nature Physics* 8, 25–31 (2011)
3. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. *Physical Review E* 69, 066133 (2004)
4. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. *Phys. Rev. E* 70, 066111 (2004)
5. Tsuchiura, H., Ogata, M., Tanaka, Y., et al.: Electronic states around a vortex core in high-Tc superconductors based on the t-J model. *Phy. Rev. B* 68(1), 012509 (2003)
6. Newman, M.E.J.: The structure and function of complex networks. *SIAM Review* 45(2), 167–256 (2003)
7. Pujol, J.M., Bejar, J., Delgado, J.: Clustering algorithm for determining community structure in large networks. *Phys. Rev. E* 77(9), 016107 (2006)
8. White, S., Smyth, P.: A spectral clustering approach to finding communities in graphs. In: Proceedings of the Fifth SIAM International Conference (2005)
9. Luxburg, U.: A tutorial on Spectral clustering. *Statistics and Computing* 17(4), 395–416 (2007)
10. Meila, M., Shi, J.: A random walks view of Spectral segmentation. *AISTATS* (2001)
11. Azran, A., Ghahramani, Z.: Spectral Methods for Automatic Multiscale Data Clustering. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (2006)
12. Yan, X., Zhu, Y., Rouquier, J.-B., Moore, C.: Active learning for node classification in assortative and disassortative networks. In: Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (Association of Computing Machinery) (2011)

Distance-Based Feature Selection from Probabilistic Data

Tingting Zhao^{1,2}, Bin Pei^{1,2}, Suyun Zhao¹, Hong Chen^{1,2}, and Cuiping Li^{1,2}

¹ Key Lab of Data Engineering and Knowledge Engineering of Ministry of Education, China

² Department of Computer Science, Renmin University of China, China

ttdejiaoluo@yahoo.com.cn,

{pei_ice, zhaosuyun, chong, licuiping}@ruc.edu.cn

Abstract. Feature selection is a powerful tool of dimension reduction from datasets. In the last decade, more and more researchers have paid attentions on feature selection. Further, some researchers begin to focus on feature selection from probabilistic datasets. However, in the existing method of feature selection from probabilistic data, the distance hidden in probabilistic data is neglected. In this paper, we design a new distance measure to select informative feature from probabilistic databases, in which both the distance and randomness in the data are considered. And then, we propose a feature selection algorithm based on the new distance and develop two accelerative algorithms to boost the computation. Furthermore, we introduce a parameter into the distance to reduce the sensitivity to noise. Finally, the experimental results verify the effectiveness of our algorithms.

Keywords: feature selection, randomness, distance, probabilistic data.

1 Introduction

Probabilistic data exist in many fields, including spatial databases, biology information system, sensor network and so on. A number of factors, such as data collection, data transformation and data reservation, contribute to the randomness of data [2, 5, 6]. Now many methods have been proposed to mine knowledge from probabilistic data. For example, possible world as a powerful tool has been used in dealing with many kinds of probabilistic data mining [1]. Unfortunately, the scale of possible worlds seriously limits its feasibility. Therefore, there is still a challenge to handle traditional mining problems in the field of probabilistic data mining.

Some researchers have done some interesting exploratory works on probabilistic data mining. For example, Smith [7] proposed an uncertain decision tree by considering both probabilistic information and the difference between real values. Based on the same goal, [8] proposed a new Naïve Bayes classification for probabilistic data by redesigning kernel function, which can deal with any type of probabilistic function. Such two methods worked on probabilistic data well, but it is regretted that these existing methods cannot effectively handle the datasets with high dimensions [1]. Therefore, it is necessary to do some preprocessing work by reducing the dimension of the probabilistic datasets. What is more, these existing works only consider the randomness of real value, but they neglect the distance of the real values.

Feature selection is a powerful tool to reduce redundant dimensions and select informative feature. The key to do feature selection is how to eliminate the irrelevant and redundant features. The irrelevant feature is used to indicate a feature with lower or no relevance to the class. The redundancy means some features can be omitted or replaced by other features without information loss. Up to now, little study has been performed on handling feature selection, considering both relevance and relativity at the same time. Such as feature ranking method [3], it only returns the features with maximal relevance and ignores relativity among the features.

This paper has made the following contributions. 1) We propose a distance-based information measure model. 2) We design a novel feature selection method, which deletes irrelevant as well as redundant features.

2 A New Distance Measure on Probabilistic Data

In this section, we first present a novel probabilistic data model used in our paper, and then give a new definition of distance on such data.

The main symbols used in this paper are listed in Table 1.

Table 1. Symbols

Symbol	Definition and Description	Symbol	Definition and Description
C	a set of condition features	R	an feature subset of C
D	a set of decision features	DP(R)	discernible power of R
U	a nonempty set of objects	DS	DS = (U, C ∪ D), a decision system

2.1 Probabilistic Data

The problem of modeling probabilistic data has been studied in many researches [4]. In this paper, we focus on a kind of feature level uncertainty, which allows every value, presented by an interval, of the same feature in different tuples has its own probability distribution. Such as the error model for values collected from sensors is assumed to be Gaussian distribution. For example, the value measured by a sensor could be modeled as $A:[60, 80]$ with a Gaussian distribution $\mathcal{N}(70, 10^2)$.

When a numerical feature is represented as an interval with an independent probability distribution, we call it random value feature (RVF), denoted by A^u . Further, we use A_i^u to denote the i th instance of A^u and T_i to denote the i th tuple in the dataset. One tuple here consists of random value features A^u and class D.

2.2 The New Distance Measure

In this subsection, we discuss how to measure the distance of such probabilistic data.

Definition 1. The distance $F(A_i^u, A_j^u)$ between A_i^u, A_j^u . Suppose pdf of A_i^u, A_j^u is $f(A_i^u), g(A_j^u)$ and range of A_i^u, A_j^u is $[a, b], [c, d]$. $F(A_i^u, A_j^u) = \frac{\left| \int_c^d \int_a^b [xf(A_i^u) - yg(A_j^u)] dx dy \right|}{\int_c^d \int_a^b dx dy}$, where x, y are random variables of A_i^u, A_j^u , and $F(A_i^u, A_j^u) \in [0, 1]$.

In fact, any type of distance formula can be used to construct the distance metric, as long as it satisfies non-negative, symmetric and triangle inequality characteristics.

Definition 2. The distance of tuples T_i and T_j is $\text{Dis}_C(T_i, T_j) = \max_{A^u \in C} \{F(A_i^u, A_j^u)\}$.

Definition 3. The discernible distance $\underline{\text{Dis}}(T_i)_D = \min(1, \text{Dis}_C(T_i, T_j) + D(T_j))$, where $D(T_j) = 1$ if T_i and T_j are in the same class, else $D(T_j) = 0$.

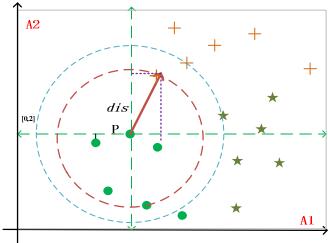


Fig. 1. The discernible distance

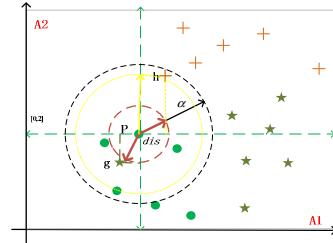


Fig. 2. α discernible distance

Figure 1 demonstrates the definition of discernible distance $\underline{\text{Dis}}(T_i)_D$. Three class tuples are drawn in different shapes, and each point is a tuple. A1 and A2 represent two condition features. dis represents the discernible distance of tuple P. The discernible distance is a minimum distance to distinguish P from tuples with different classes.

3 Feature Selection in Probabilistic Database

In this section, a new way to measure the information hidden in probabilistic databases is proposed. This measure is defined based on discernible distance. And then, we introduce a novel feature selection method of probabilistic databases. In our paper, the information means the discernible power in probabilistic databases.

3.1 Distance-Based Information Measure

Definition 4 (Discerned-Pair). If a pair $(T_i, T_j) \in U \times U, i \neq j$ and $D(T_i) \neq D(T_j)$, (T_i, T_j) is called a discerned-pair of dataset U .

Definition 5 (Discernible Space). The discernible space of feature subset R satisfies $DSp(R) = \{(T_i, T_j) | \text{Dis}_R(T_i, T_j) \geq \underline{\text{Dis}}(T_i)_D\}$, where $T_i, T_j \in U$, and $R \subseteq C$.

So Discerned-Pair represents the sample pair with different decision class, and discernible space of R consists of all discerned-pairs distinguished by feature set R.

Definition 6 (Discernible Power of R). The discernible power of R is defined as the cardinality of discernible space $DSp(R)$: $DP(R) = |DSp(R)|$.

Definition 6 shows $DP(R)$ can measure the ability of R to discern each pair in a non-empty finite set of objects. Meanwhile, the larger $DP(R)$ is, the greater discernible power R. In a sense, $DP(R)$ can be seen as a measure for information hidden in probabilistic data, and $DP(C)$ holds all information hidden in probabilistic data.

3.2 Distance-Based Feature Selection

In the above subsection, $DP(R)$ can be as a measure of information hidden in probabilistic data. Hence, if we select an appropriate subset B and keep $DP(B) = DP(C)$, the information used to classify different class tuples keeps unchanged. In general, we can select the features, which is with maximal discernibility and minimal relativity.

When we just find a subset with the maximal discernibility, the relativity among the features still exists. For example, a feature ranking method [3] selects two features both with bigger discernibility, but they might distinguish almost the same discerned-pairs. Here redundancy happens. To solve such problem, the concept of minimal relativity is mentioned. Minimal relativity between features mainly embodies in the maximal difference of their discerned-pair space. In general, we can select the features, which have the higher discernible power as different discernible space as possible.

4 Algorithms to Feature Selection

4.1 A Heuristic Algorithm

Generally speaking, the naïve algorithm is first to find all subsets of the original feature set and compute their own discernible power. The subset with the fewest features can be a final subset when its discernible power is equal to the original feature set. Obviously, such a naïve method is NP complete problems.

Hence, we propose a heuristic algorithm to find a final subset. The algorithm starts with an empty set, and keeps adding features into a pool until the discernible power of all features in the pool is the biggest. The algorithm is described as follows.

Algorithm 1. Feature selection with discernible power (FSDP)

```

 $RED \leftarrow \{ \}, DS = (U, C \cup D)$ 
(1):Compute discernible power of  $C:DP(C)$ ;
(2):For every  $a_i \in C$ , compute  $DP(a_i)$  ;  $RED \leftarrow a_k$ .if  $DP(a_k) = \max_{a_i \in C} DP(a_i)$ , and if  $DP(a_k) = DP(C)$ go to (5), else go to (3);
(3):Let  $S_i = RED \cup a_i, a_i \in \{C - RED\}$ , compute  $DP(S_i)$ ;  $RED \leftarrow S_k$ ,if  $DP(S_k) = \max DP(S_i)$ ;;
(4):Do (3) until  $DP(RED) = DP(C)$  ;
(5):Output  $RED$ .

```

4.2 The Accelerative Algorithm

Meanwhile, two accelerative algorithms are designed to get better performance.

The first accelerative algorithm makes full use of uniqueness to measure whether a feature has priority to be selected firstly or not. The discerned-pairs, which can only be distinguished by the feature selected in priority, are being deleted from discerned-pair space. The selection process continues until the discerned-pair space is empty. We call this accelerative algorithm discerned degree-based algorithm (FSDD).

The second accelerative algorithm is a search and delete algorithm (FSSD). It saves execution time by reducing space. First, it searches the feature with the biggest

discerned power through the original discerned-pair space, and then deletes the discerned-pairs that can be distinguished by this feature. This feature is then added to final subset and we keep searching the next feature until the reduced space is empty.

4.3 The Scalable Algorithm with a Variable

Although the accelerative algorithms improve the performance of feature selection process, they are still sensitive to noise. This problem can be analyzed with Figure 2. Point g is an object with noise. When we compute the discernible distance for object P , we get a smaller discernible distance dis represented with the red line. This smaller dis leads to an obvious question: object P is completely isolated.

To solve such problem, we redefine Definition 3, 5 with variable α . In Definition 7, through adding α , the discernible distance increases. As shown in Figure 2, by adding variable α , the effect of object P that was isolated with others could be somewhat neglected. After replacing old definitions, we get the general algorithms with variable α , which are less sensitive to noise. Here, $\alpha \in [0,1]$.

Definition 7 (α discernible distance). Redefine discernible distance $\underline{Dis}_\alpha(T_i)_D = \inf_{T_j \in U \wedge D(T_j) \leq \alpha} \min(1, Dis_C(T_i, T_j) + \alpha) \wedge \inf_{T_j \in U \wedge D(T_j) > \alpha} \min(1, Dis_C(T_i, T_j) + D(T_j))$.

Definition 8(α discernible space). we redefine the α discernible space $DSp_\alpha(R) = \{(T_i, T_j) | \max(\underline{Dis}_\alpha(T_i)_D - Dis_R(T_i, T_j), 0) \leq \alpha\}$ where $T_i, T_j \in U$, and $R \subseteq C$.

5 Experimental Results

In this section, we present the experiments to evaluate the algorithms. 4 datasets which contain numerical features are chosen from the UCI (See Table 2). The original numerical data are converted into interval values with probability distribution. The procedure is as follows: Scan each feature and get its maximum value C_{max} and minimum value C_{min} , respectively. Let range = $C_{max} - C_{min}$. In our probability model, we use Gaussian distribution over the range without loss of generality. For each feature A^j , suppose the range of A^j is $C_{max}^j - C_{min}^j$, and the original point value of tuple T_i is A_i^j , so A^j in T_i obeys to Gaussian distribution $(A_i^j, (w\% * (C_{max}^j - C_{min}^j))^2)$, where w is a percentage parameter that can control randomness degree of the objects.

Table 2. Datasets

Symbol	Datasets	Objects	Features	Decision classes
D1	Parkinsons	195	22	2
D2	Wdbc	569	23	2
D3	Movement_libras	360	90	15
D4	ConnectionistBench	208	60	2

There are two factors to measure the performances of our algorithms. One is the size of the final subset, and the other is the execution time. In addition, there are two main parameters α and w in our algorithms.

5.1 Effect of w

When we increase the value of w , the execution time and the number of feature subset of our three algorithms are recorded in Figure 3. Since datasets D1-D4 have similar results, we only take *parkinsons* as an example and fix $\alpha=0.9$.

Overall, the execution time is positively related to the scale of the subset. If the scale of subset keeps unchanged, the execution time is basically unchanged. The randomness reacts on two aspects. One is the distance among data, and the other is the discernibility of features. Effect of randomness on different algorithms in the former aspect has the homologous results. In the latter aspect, the changed discernibility of features lead to the scale of the feature subset varies to a certain extent.

5.2 Effect of α

The parameter α is used to decrease the effect of noise. To avoid the effect of w , we fix $w = 0.05$. In Figure 4, each subgraph draws the execution time with the value of α from 0.9 to 0.99. The final size of a feature subset is recorded in Table 3. From the table, we can see that our algorithms decrease the scale of features effectively.

Table 3. The Scale of Feature Subset VS α

Algorithms/ α	Datasets	0.9	0.94	0.98	0.99	Datasets	0.9	0.94	0.98	0.99
FSDP	D1	9	9	7	6	D3	48	48	47	48
		10	6	4	3		14	8	5	4
		9	9	7	6		50	50	50	50
FSSD	D2	11	11	11	10	D4	41	42	40	40
		12	10	6	5		9	7	7	7
		11	11	11	10		53	53	53	53

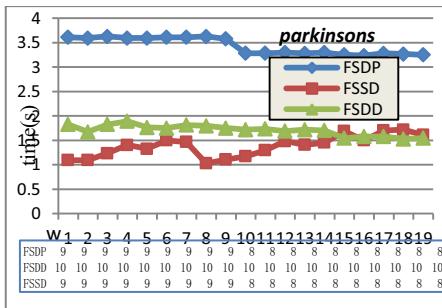


Fig. 3. The effect of w

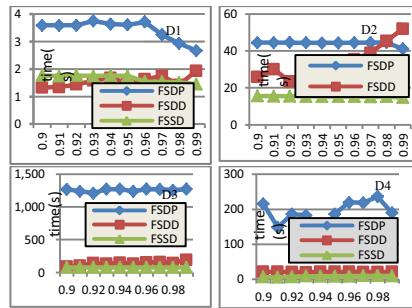


Fig. 4. The execution time VS α

Figure 4 shows FSSD mostly costs the fewest execution time among the three algorithms no matter how the value of α changes. Meanwhile, Table 3 shows that its ability to find a smaller subset weakens when datasets with more features. On the other hand, FSDD is powerful to decrease the size of the feature subset when α increases. This phenomenon can be explained from the view of the parameter α , which

reduces the effect of noise to keep the discernibility of features. So FSDD makes full use of such discernibility and finally selects a smaller subset. Meanwhile, FSSD has the fewest execution time whereas FSDD is good at selecting a smaller feature subset.

6 Conclusion

Probabilistic data models have been discussed in many applications, including spatial databases, sensor networks and biology information systems. However, less effort has been put on the feature selection from probabilistic data considering both randomness and distance. In this paper, we propose a new distance-based feature selection method.

Acknowledge. This work was supported by the National Science Foundation of China (Grant No.612 02114, 61070056,61033010, 61272137).

References

1. Aggarwal, C.C., Yu, P.S.: A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering* 21(5) (May 2009)
2. Hall, M.: Correlation-based feature selection for discrete and numeric class machine learning. In: Pro. 17th Int'l Conf. Machine Learning (2000)
3. Dash, M., Liu, H.: Consistency-based search in feature selection. *Artificial Intelligence* 151, 155–176 (2003)
4. DasSarma, A., Ben, O., Halevy, A., Widom, J.: Working models for uncertain data. In: ICDE (2006)
5. Ngai, W., Kao, B., Chui, C., Cheng, R., Chau, M., Yip, K.Y.: Efficient Clustering of Uncertain Data. In: Proc. Sixth IEEE Int'l Conf. Data Mining, ICDM (2006)
6. Srikant, R., Agrawal, R.: Mining generalized association rules. VLDB (1995)
7. Tsang, S., Kao, B., Yip, K.Y., Ho, W.-S., Lee, S.D.: Decision trees for uncertain data. TKDE (2011)
8. Ren, J., Lee, S.D., Chen, X., Kao, B., Cheng, R., Cheung, D.: Naïve Bayes Classification of Uncertain Data. In: ICDM 2009 (2009)

Aspect-Specific Polarity-Aware Summarization of Online Reviews

Gaoyan Ou^{1,2}, Wei Chen^{1,2}, Peng Liu^{1,2}, Tengjiao Wang^{1,2},
Dongqing Yang^{1,2}, Kai Lei³, and Yueqin Liu⁴

¹ Key Laboratory of High Confidence Software Technologies,
Ministry of Education, Beijing 100871, China

² School of Electronics Engineering and Computer Science, Peking University,
Beijing 100871, China

³ The Shenzhen Key Lab for Cloud Computing Technology and Applications
(SPCCTA), Peking University Shenzhen Graduate School, Shenzhen 518055, China

⁴ Department of Information Science and Technology,
University of International Relations, Beijing, China
pekingchenwei@pku.edu.cn

Abstract. With the popularity of various social media platforms, the number of online reviews towards different products and services grows dramatically. Discovering sentiments from online reviews becomes an important and challenging task in sentiment analysis. Current methods either extract aspects without separating aspects and sentiments, or extract aspects and sentiments without separating sentiments according to their polarities. In this paper, we propose two novel probabilistic generative models (APSM and ME-APSM) to extract aspects and aspect-specific polarity-aware sentiments from online reviews. We applied our models to two data sets with three different experiments. Experimental results show that APSM and ME-APSM models can extract aspects and polarity-aware sentiments well. For the sentiment classification task, our models outperform other generative models and come close to supervised classification methods.

Keywords: aspect extraction, topic modeling, sentiment analysis, review analysis.

1 Introduction

With the rapid growth of Web 2.0, there exist a large amount of reviews on various types of social medias, such as discussion forums, microblogs, blogs and shopping websites. These reviews contain people's opinions and sentiments towards different products and services. It is important and helpful for both customers and manufacturers to discover and summarize aspects and sentiments from online reviews in detail. It is still a challenging task for the following two reasons. First, it is almost useless to do analysis manually because of the huge number of reviews. Second, the reviews are composed of unstructured texts. Accurately understanding these texts is difficult for machines.

Most earlier researches [1–3] adopt supervised learning models to classify the entire document as positive or negative. However, sentiment polarities are often dependent on topics or aspects. For example, in the hotel review “The staff are very friendly, but the price is so high!”, sentiment for aspect *staff* is positive while sentiment for aspect *price* is negative. It is not sufficient to simply classify the entire review as positive or negative. Therefore, it is more suitable to analyze aspect and sentiment simultaneously.

In recent years, how to automatically discover aspects and sentiments from online reviews has attracted many attentions. Recent researches have proposed many methods to extract aspects and sentiments from online reviews. These methods can be divided into two categories. One kind of method only extracts aspects without extracting sentiments. However, it can not provide aspect-specific sentiment information, which is also very important. The other kind extracts aspects and sentiments simultaneously. This kind of method first identifies different aspects for the given product and then extracts specific sentiments for each aspect. In the extracted sentiments for each aspect, positive and negative words are mixed together. This makes it hard for users to know how sentiment are expressed according to different polarities for a particular aspect.

In this paper, we focus on the problem of simultaneously aspect and sentiment extraction and sentiment classification of online reviews. The problem setup is illustrated in Fig. 1. We propose two novel probabilistic generative models to address this problem. The first model is called APSM and the second model is called ME-APSM. Our models can not only extract different aspects (e.g. *staff*, *room*, *meal* and *price* for hotel), but also extract specific sentiments (positive or negative) for each aspect.

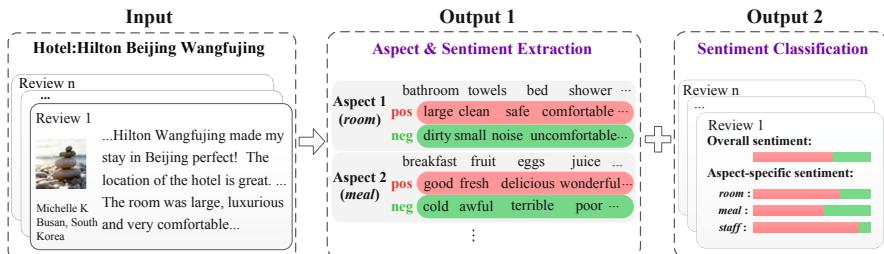


Fig. 1. Problem Setup

Compared to existing methods, the key advantage of our models is that we can extract polarity-aware sentiments for each aspect. For example, for aspect *room*, positive sentiments are “large”, “clean” and “safe” while negative sentiments are “dirty”, “small” and “noise”. Another advantage of APSM and ME-APSM is that they can be directly applied to the sentiment classification task for the entire review. Moreover, it is easy to integrate aspect and sentiment prior knowledge into our models.

The rest of the paper is organized as follows. Section 2 introduces the related work. In section 3, we present our models and describe how to integrate prior information. Section 4 describes the data sets and experiment settings. Section 5 shows our experiment results. Section 6 concludes the research and gives directions for future studies.

2 Related Work

There are many existing works on aspect-based sentiment analysis. One approach is to use frequent itemset mining algorithm to extract frequent nouns and noun phrases as aspect candidates [4–6]. The main limitations of frequent-based methods are that they do not group related aspects together and can not extract implicit aspect expressions. Sequential labeling techniques are also used to extract aspects and sentiments from reviews [7–9]. These supervised methods suffer from the hardness to obtain labeled training data.

In recent years, several unsupervised unified aspect and sentiment models have been proposed. Zhao [10] proposed ME-LDA model by using a MaxEnt component to help separate aspect and sentiment words. ME-SAS proposed in [11] can further integrate aspect seeds by introducing a two-level tree structured aspect distribution. Both ME-LDA and ME-SAS extract aspects by separating aspect and sentiment words. However, they do not separate sentiments according to their polarities.

Some researchers focus on the separating of different sentiment polarities [12–14]. Lin and He [12] presented a JST model which can detect review-level sentiment and extract mixture of aspects. The ASUM model proposed in [14] improved JST by constraining the words in a single sentence to come from the same language model. However, aspect and sentiment words are mixed together in the aspects extracted by JST and ASUM, which make it hard for users to understand.

Our proposed APSM and ME-APSM models can not only separate aspect and sentiment words for each extracted aspect, but also separate sentiment words according to their polarities. Moreover, APSM and ME-APSM are capable of detecting review-level sentiments.

3 The Proposed Models

In this section, we first present our proposed Aspect-specific Polarity-aware Sentiment model (APSM). Then we extend APSM to *Maximum Entropy Aspect-specific Polarity-aware Sentiment Model* (ME-APSM), with a maximum entropy component. Finally we describe how to integrate sentiment and aspect prior information to the proposed models by using asymmetric Dirichlet prior.

3.1 APSM Model

To understand how we model the reviews, we demonstrate the generative process of APSM by the following scenario. The reviewer wants to write a review

about a hotel. She first decides a distribution of aspects, for example, 30% about *price*, 40% about *staff* and 30% for other aspects. Then she decides a sentiment distribution for each aspect, for example, 80% satisfied and 20% unsatisfied for aspect *price*. For each sentence, she decides which aspect to evaluate and what sentiment to express about the aspect. She also decides the distribution of aspect and sentiment of the sentence. For each word in a sentence, she first decides whether it is an aspect word or a sentiment word. If she chooses aspect, then the word is generated from the aspect model. If she chooses a sentiment, then the word is generated from the aspect-specific sentiment model.

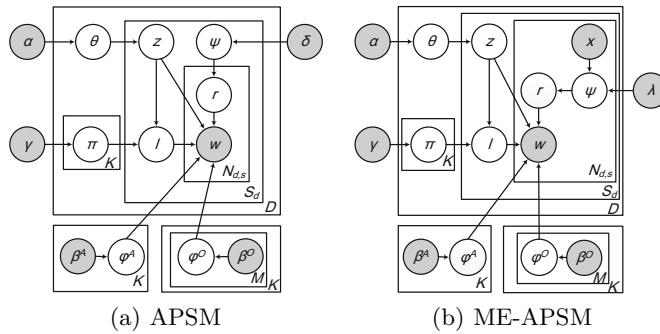


Fig. 2. Graphical Representation of APSM and ME-APSM

The graphical representation of APSM is shown in Fig.2(a). Let V be the vocabulary size, D be the number of reviews, S_d be the number of sentences in review d and $N_{d,s}$ be the number of words in sentence s of review d . Let K denote the number of aspects and M denote the number of sentiments. There are K aspect models $\varphi_{k=1 \dots K}^A$. For each aspect k , there are M aspect-specific sentiment models $\varphi_{k,m=1 \dots M}^O$. We further introduce a switch variable $r \in \{\hat{a}, \hat{o}\}$ for each word to denote whether it is an aspect word or a sentiment word. The variable $\psi_{d,s}$ denotes the distribution of aspects and sentiments in sentence s of review d .

The formal generative process of APSM is as follows:

1. For each aspect $k \in \{1, \dots, K\}$:
 - i. Draw $\varphi_k^A \sim Dir(\beta_k^A)$
 - ii. For each sentiment $m \in \{1, \dots, M\}$:
Draw $\varphi_{k,m}^O \sim Dir(\beta_{k,m}^O)$
2. For each review $d \in \{1, \dots, D\}$:
 - i. Draw $\theta_d \sim Dir(\alpha)$:
 - ii. For each aspect $k \in \{1, \dots, K\}$:
Draw $\pi_{d,k} \sim Dir(\gamma)$
 - iii. For each sentence $s \in \{1, \dots, S_d\}$:
 - (a) Draw $z_{d,s} \sim Mult(\theta_d)$

- (b) Draw $l_{d,s} \sim Mult(\pi_{d,z_{d,s}})$
- (c) Draw $\psi_{d,s} \sim Beta(\delta)$:
- (d) For each term $w_{d,s,n}$ where $n \in \{1, \dots, N_{d,s}\}$:
 - I. Draw $r_{d,s,n} \sim Bernoulli(\psi_{d,s})$, $r_{d,s,n} \in \{\hat{a}, \hat{o}\}$
 - II. if $r_{d,s,n} = \hat{a}$ // $w_{d,s,n}$ is an aspect
 Draw $w_{d,s,n} \sim Mult(\varphi_{z_{d,s}}^A)$
 else draw $w_{d,s,n} \sim Mult(\varphi_{z_{d,s}, l_{d,s}}^O)$

We use collapsed Gibbs Sampling [15] to inference the model. Due to space limit, we only show the sampling formulas without detailed derivations. The sampling formula of latent variables z and l is:

$$p(z_{d,s} = k, l_{d,s} = m | Z_{\neg d,s}, L_{\neg d,s}, R, W) \propto \frac{\alpha_k + n_{d,k,\neg d,s}^{Sent.}}{\sum_k (\alpha_k + n_{d,k,\neg d,s}^{Sent.})} \frac{\gamma_m + n_{d,k,m,\neg d,s}^{Sent.}}{\sum_m (\gamma_m + n_{d,k,m,\neg d,s}^{Sent.})} \\ \frac{\Gamma(\sum_v \beta_{k,v}^A + n_{k,v,\neg d,s}^A)}{\Gamma(\sum_v (\beta_{k,v}^A + n_{k,v,\neg d,s}^A) + n_{d,s}^A)} \prod_{v=1}^V \frac{\Gamma(\beta_{k,v}^A + n_{k,v,\neg d,s}^A + n_{d,s,v}^A)}{\Gamma(\beta_{k,v}^A + n_{k,v,\neg d,s}^A)} \\ \frac{\Gamma(\sum_v \beta_{k,m,v}^O + n_{k,m,v,\neg d,s}^O)}{\Gamma(\sum_v (\beta_{k,m,v}^O + n_{k,m,v,\neg d,s}^O) + n_{d,s}^O)} \prod_{v=1}^V \frac{\Gamma(\beta_{k,m,v}^O + n_{k,m,v,\neg d,s}^O + n_{d,s,v}^O)}{\Gamma(\beta_{k,m,v}^O + n_{k,m,v,\neg d,s}^O)} \quad (1)$$

where $\Gamma(\cdot)$ is the gamma function. $n_{d,k}^{Sent.}$ is the number of sentences assigned to aspect k in review d . $n_{d,k,m}^{Sent.}$ is the number of sentences assigned to aspect k and sentiment m in review d . $n_{d,s}^{(O)}$ is the number of aspect (sentiment) words in sentence s of review d . $n_{d,s,v}^{(O)}$ is the number of times word v assigned to aspect (sentiment) word in sentence s of review d . $n_{k,v}^A$ is the number of times word v assigned to aspect k as an aspect word. $n_{k,m,v}^O$ is the number of times word v assigned to aspect k and sentiment m . $\neg d, s$ denotes counts excluding sentence s of review d .

Assume that $z_{d,s} = k$ and $l_{d,s} = m$, the sampler of latent variable r is :

$$p(r_{d,s,n} = \hat{o} | Z, L, R_{\neg d,s,n}, W) \propto (\delta_O + n_{d,s,\neg d,s,n}^O) \frac{\beta_{k,m,v}^O + n_{k,m,v,\neg d,s,n}^O}{\sum_v (\beta_{k,m,v}^O + n_{k,m,v,\neg d,s,n}^O)} \quad (2)$$

$$p(r_{d,s,n} = \hat{a} | Z, L, R_{\neg d,s,n}, W) \propto (\delta_A + n_{d,s,\neg d,s,n}^A) \frac{\beta_{k,v}^A + n_{k,v,\neg d,s,n}^A}{\sum_v (\beta_{k,v}^A + n_{k,v,\neg d,s,n}^A)} \quad (3)$$

where $\neg d, s, n$ denotes counts excluding $w_{d,s,n}$.

With the above samples, we can estimate the model parameters θ , π , φ^A and φ^O as:

$$\theta_{d,k} = \frac{n_{d,k}^{Sent.} + \alpha_k}{\sum_{k=1}^K (n_{d,k}^{Sent.} + \alpha_k)} \quad (4)$$

$$\pi_{d,k,m} = \frac{\sum_{m=1}^M (n_{d,k,m}^{Sent.} + \gamma_m)}{\sum_{m=1}^M (n_{d,k,m}^{Sent.} + \gamma_m)} \quad (5)$$

$$\varphi_{k,v}^A = \frac{n_{k,v}^A + \beta_{k,v}^A}{\sum_{v=1}^V (n_{k,v}^A + \beta_{k,v}^A)} \quad (6)$$

$$\varphi_{k,m,v}^O = \frac{n_{k,m,v}^O + \beta_{k,m,v}^O}{\sum_{v=1}^V (n_{k,m,v}^O + \beta_{k,m,v}^O)} \quad (7)$$

We use $\theta_{d,k}$ to estimate the probability of aspect k in review d . We extract the aspect words for each aspect by $\varphi_{k,v}^A$. $\varphi_{k,m,v}^O$ is used to identify aspect-specific sentiment words. $\pi_{d,k,m}$ is used to analyze the sentiment distribution of aspect k in review d . The overall sentiment score of entire review d is estimated as follows:

$$Score(d, m) = \sum_{k=1}^K \theta_{d,k} \cdot \pi_{d,k,m} \quad (8)$$

3.2 ME-APSM Model

We improve APSM by employing Maximum Entropy (MaxEnt) priors to separate sentiment and aspect words better. This method was previously used in [10] and [11]. The basic idea is that aspect words tend to be nouns or noun phrases, while sentiment words tend to be adjective and adverbs. Thus we can train a MaxEnt by utilizing the POS features to help separate sentiments and aspects.

Following [10] and [11], we choose two types of features: lexical features and POS features. For word $w_{d,s,n}$, we denote its corresponding feature vector as $\mathbf{x}_{d,s,n} = \{w_{d,s,n-1}, w_{d,s,n}, w_{d,s,n+1}, POS_{d,s,n-1}, POS_{d,s,n}, POS_{d,s,n+1}\}$. The MaxEnt can be trained automatically by using a sentiment lexicon, eliminating the need of manual annotation of labeled training data. We indicate the new model as ME-APSM, as shown in Fig.1(b).

In ME-APSM, the Gibbs samplers are the same except for variable r . The sampler for r changes as:

$$p(r_{d,s,n} = \hat{o}|Z, L, R_{\neg d,s,n}, W) \propto \exp\left(\sum_i \lambda_i f_i(x_{d,s,n}, \hat{o})\right) \frac{\beta_{k,m,v}^O + n_{k,m,v \neg d,s,n}^O}{\sum_v (\beta_{k,m,v}^O + n_{k,m,v \neg d,s,n}^O)} \quad (9)$$

$$p(r_{d,s,n} = \hat{a}|Z, L, R_{\neg d,s,n}, W) \propto \exp\left(\sum_i \lambda_i f_i(x_{d,s,n}, \hat{a})\right) \frac{\beta_{k,v}^A + n_{k,v \neg d,s,n}^A}{\sum_v (\beta_{k,v}^A + n_{k,v \neg d,s,n}^A)} \quad (10)$$

where $f_{1\dots n}$ are the n binary features of the learned MaxEnt model and $\lambda_{1\dots n}$ are their corresponding weight parameters.

3.3 Incorporating Prior Knowledge

Although APSM and ME-APSM are unsupervised models, we can incorporate prior information which can be obtained in many ways. We consider the following two kinds of prior information: sentiment prior and aspect prior.

Sentiment Prior. The first type of sentiment prior can be obtained from sentiment lexicon. In some cases, user can also provide a few positive and negative sentiment seeds for some aspects. We incorporate these two types of sentiment prior information into hyperparameter β^O .

Let $M = \{M_p, M_n\}$ denote the sentiment lexicon, where M_p is the positive word list and M_n is the negative word list. Let $\Omega_{k,m}^O$ denote the aspect sentiment seeds. Intuitively, we expect that no negative sentiment word appears in each aspect's positive sentiment model, and vice versa. We also expect that positive (negative) word will be more likely to appear in each aspect's positive (negative) sentiment model. Under these intuitions, we define β^O as follows:

$$\beta_{k,m='pos',v}^O = \begin{cases} 0 & \text{if } v \in M_n, \\ 0.1 & \text{if } v \in M_p, \\ 0.2 & \text{if } v \notin M_p \text{ and } v \in \Omega_{k,'pos}^O, \\ 0.01 & \text{others.} \end{cases} \quad (11)$$

Aspect Prior. In many cases, user can provide some seed words for a few aspect categories. We integrate this information into the hyperparameter β^A . For aspect k , say user provides a seed set Ω_k , we define β_k^A as follows. If word v is a sentiment word, then $\beta_{k,v}^A = 0$; if word v is in the seed set Ω_k , then we set $\beta_{k,v}^A = 0.1$; else $\beta_{k,v}^A = 0.01$.

4 Experimental Setup

We use two data sets to evaluate our proposed models. The first data set provided in [16] contains 10508 hotel reviews from TripAdvisor¹. We construct a balanced version of this data set, which contains 2500 positive reviews and 2500 negative reviews. The second data set is a Amazon product reviews data set from [17]. This data set contains four product types: books, DVDs, electronics and kitchen appliances. For each product type, there are 2000 positive reviews and 2000 negative reviews.

To learn the parameters $\lambda_{1\dots n}$ of ME-APSM, we randomly select 2000 sentences from both data sets. Then we use Stanford POS Tagger² to tag the reviews. Words contained in the sentiment lexicon are automatically labeled as sentiment words, otherwise as aspect words. We use the sentiment lexicon from [4], which contains 2006 positive sentiment words and 4783 negative sentiment words. The sentiment lexicon is also used to incorporate prior information into APSM and ME-APSM, as described in section 3.3.

In our experiments, the number of aspects T is set to be 30 and the number of sentiments M is set to be 2. For all models, we set the Gibbs sampling iterations to be 5000. We fix α and γ as: $\alpha = 50/T$, $\gamma = 1$. Following [11], we set $\delta_A = 2.35$, $\delta_O = 3.44$ in APSM.

¹ <http://www.tripadvisor.com/>

² <http://nlp.stanford.edu/software/tagger.shtml>

5 Experiments

In this section, we evaluate the performances of our proposed models with three experiments. In the first experiment, we show the aspects and aspect-specific sentiments extracted by APSM and ME-APSM with some qualitative analysis. The second experiment evaluates sentiment identification performances of our models. In the third experiment, we apply a review-level sentiment classification task to compare our models with several baselines.

5.1 Qualitative Results

In the first experiment, we show some sample aspects and aspect-specific sentiments extracted by APSM and ME-APSM by using the TripAdvisor data set. Table 1 lists the top 10 aspect words of three aspects (*staff*, *room* and *meal*) discovered by APSM and ME-APSM. For each aspect, top 10 positive and top 10 negative sentiment words are also listed.

We can see from Table 1 that both APSM and ME-APSM can extract coherent aspects and aspect-specific sentiments well. For example, “breakfast”, “coffee”, “buffet”, “fruit” and “eggs” are all words related to the aspect *meal*. They are correctly identified by APSM and ME-APSM. In general, ME-APSM performs better than APSM. For the aspect *staff*, both APSM and ME-APSM discover aspect word “staff”, but APSM fails to discover more staff-related words like “waiter” and “waitress”, which are successfully captured by ME-APSM. APSM incorrectly identifies the aspect word “staff” as positive sentiment words. ME-APSM can discover more specific negative sentiment words, such as “rude” and “unfriendly”.

5.2 Aspect-Specific Sentiment Extraction

The key advantage of APSM and ME-APSM is their ability to identify polarity-aware sentiment words for different aspects. In this second experiment, we quantitatively evaluate the quality of positive and negative sentiment words identified by our models. The metric precision at n ($P@n$) is used for evaluation. We use the manually annotated aspect-specific sentiment gold standard from [18]. Since there is no aspect-specific sentiment gold standard for the product data set, we only use the hotel data set.

We choose ME-LDA [10] as the compared method. ME-LDA can also extract aspects and sentiments simultaneously. However, it does not directly separate positive and negative sentiment words. Thus, we apply the following postprocessing process for ME-LDA. Given the ranked sentiment words for each aspect, we obtain the positive word list by removing negative words; we obtain the negative word list by removing positive words.

Table 2 gives $P@n$ values for ME-LDA, APSM and ME-APSM. Due to space limit, we only show the results on three aspects (*staff*, *room* and *meal*). It can be seen that APSM and ME-APSM give better results than ME-LDA. ME-APSM further outperforms APSM, which suggests the effectiveness of the MaxEnt component.

Table 1. Example Aspects and Sentiments Extracted by APSM and ME-APSM

Aspect	APSM			ME-APSM		
	Aspect	Senti(p)	Senti(n)	Aspect	Senti(p)	Senti(n)
Staff	staff helpful friendly english desk front good extremely nice spoke	staff friendly courteous helpful attentive clean great recommend good nice	unhelpful poor bad noise cold problem overpriced disappointed bother arrogant	staff helpful friendly english desk extremely warter waitress breakfast spoke	good great helpful friendly excellent wonderful staff clean efficient pleasant	rude unfriendly unhelpful noise poor disappointed cheap hard grumpy complaints
Room	shower room bathroom water small bath clean towels hot good	nice clean great spacious comfortable good modern safe warm free	smell dirty stall cramped cold problem drain broken worn problems	room bathroom clean bed shower small comfortable large size spacious	rooms large good huge comfortable shower nice great clean room	small hard tired uncomfortable noise bad worn cold broken dark
Meal	breakfast coffee buffet room fruit eggs fresh included great continental	breakfast friendly fresh variety good great delicious nice clean hot	cold scrambled problem hard bad expensive poor die cheap miss	breakfast coffee fruit buffet eggs cheese cereal juice fresh pastries	good great fresh hot wonderful excellent nice fantastic decent loved	cold scrambled awful limited terrible bad poor disappointed cheap negative

Table 2. Aspect-specific Sentiment Extraction Performance

Aspect/Sentiment	ME-LDA			APSM			ME-APSM		
	P@5	P@10	P@20	P@5	P@10	P@20	P@5	P@10	P@20
Staff/Pos	1.00	0.90	0.65	0.80	0.70	0.70	1.00	0.80	0.80
Staff/Neg	0.40	0.60	0.35	0.80	0.50	0.35	0.80	0.40	0.30
Room/Pos	0.80	0.60	0.70	1.00	0.90	0.80	1.00	0.80	0.75
Room/Neg	0.60	0.30	0.25	0.40	0.50	0.30	0.80	0.50	0.40
Meal/Pos	0.80	0.80	0.70	0.80	0.80	0.85	1.00	0.80	0.85
Meal/Neg	0.20	0.30	0.30	0.40	0.30	0.35	0.60	0.40	0.30
Avg./Pos	0.87	0.77	0.68	0.87	0.80	0.78	1.00	0.80	0.80
Avg./Neg	0.40	0.40	0.30	0.53	0.43	0.35	0.73	0.43	0.33

5.3 Sentiment Classification

In this section, we present the results of sentiment classification. We compare the performance of our models with ASUM [14], lexicon-based method and supervised method [19]. The lexicon-based method classifies each review according to the number of positive and negative words contained in the review.

The experimental results are shown in Table 3. The performance of unified aspect and sentiment models (ASUM, DS-LDA, APSM and ME-APSM) are better than lexicon based method on both data sets. This is because sentiment polarities are dependent on aspects. The lexicon based method can not capture the aspect information of the sentiment words. ASUM and our models jointly model aspect and sentiment, which can improve the sentiment classification accuracy.

APSM and ME-APSM consistently outperform ASUM. ASUM can not separate aspects and sentiments. This suggests that separating aspects and sentiments not only improve the aspect extraction performance, but also improve sentiment classification accuracy.

To study the effect of aspect and sentiment seeds on sentiment classification. We choose three aspect seeds for three aspects. For each aspect, we choose three positive seeds and three negative seeds. We indicate the models with aspect and sentiment seeds as APSM+ and ME-APSM+. It can be seen from Table 3 that APSM+ performs better than APSM and ME-APSM+ performs better than ME-APSM. The performace of ME-APSM+ is comparable to supervised method in [19]. Note that the supervised method needs labeled training data while ME-APSM+ only needs several aspect and sentiment seeds. This suggests that incorporating aspect and sentiment seeds can improve sentiment classification performance.

Table 3. Sentiment Classification Results

Method	Hotel Data Set	Product Data Set
Lexicon-based Method	62.7%	60.2%
ASUM	65.6%	64.5%
APSM	69.7%	66.5%
ME-APSM	72.9%	69.2%
APSM+	70.3%	66.9%
ME-APSM+	73.9%	70.1%
Supervised Classification	74.3%	70.7%

We also analyze the influence of the number of aspects T . The experimental results are shown in Fig. 3. We can see from Fig. 3 that as the number of aspects increase, the sentiment classification performance increases. This trend is more evident on the product data set.

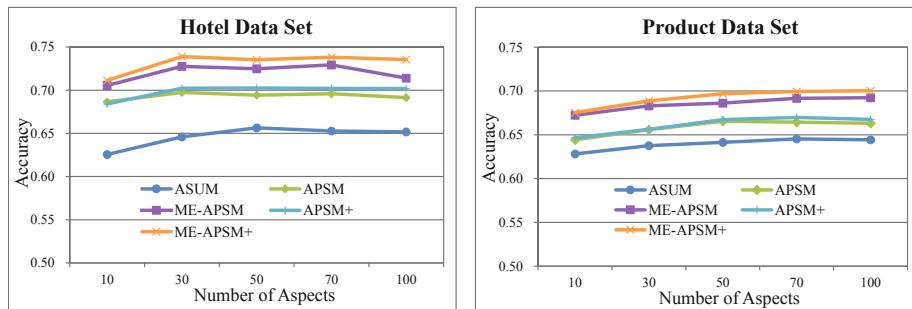


Fig. 3. Sentiment Classification Accuracy with Different Aspect Numbers

6 Conclusion

In this paper, we focus on the problem of simultaneously aspect and sentiment extraction and sentiment classification of online reviews. We proposed two unified aspect and sentiment models (APSM and ME-APSM) to address the problem. Our models can not only extract aspects and polarity-aware sentiments for each aspect, but also be applied to the sentiment classification task. We compared our models against existing approaches using three different experiments. The experiments give promising results.

In APSM and ME-APSM, how to improve the sentiment classification performance still needs more work. In the future, we also plan to apply our models to more sentiment analysis tasks, such as aspect-level sentiment classification.

Acknowledgments. This research is supported by the National High Technology Research and Development Program of China (Grant No. 2012AA011002, 2011AA010706).

References

- Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. CoRR cs.CL/0205070 (2002)
- Mao, Y., Lebanon, G.: Isotonic conditional random fields and local sentiment flow. In: NIPS, pp. 961–968 (2006)
- Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL (2005)
- Hu, M., Liu, B.: Mining and summarizing customer reviews. In: KDD, pp. 168–177 (2004)
- Popescu, A.M., Nguyen, B., Etzioni, O.: Opine: Extracting product features and opinions from reviews. In: HLT/EMNLP (2005)
- Moghaddam, S., Ester, M.: Opinion digger: an unsupervised opinion miner from unstructured product reviews. In: CIKM, pp. 1825–1828 (2010)

7. Jin, W., Ho, H.H.: A novel lexicalized hmm-based learning framework for web opinion mining. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 465–472. ACM, New York (2009)
8. Jakob, N., Gurevych, I.: Extracting opinion targets in a single and cross-domain setting with conditional random fields. In: EMNLP, pp. 1035–1045 (2010)
9. Choi, Y., Cardie, C.: Hierarchical sequential learning for extracting opinions and their attributes. In: ACL (Short Papers), pp. 269–274 (2010)
10. Zhao, W.X., Jiang, J., Yan, H., Li, X.: Jointly modeling aspects and opinions with a maxent-lda hybrid. In: EMNLP, pp. 56–65 (2010)
11. Mukherjee, A., Liu, B.: Aspect extraction through semi-supervised modeling. In: ACL (1), pp. 339–348 (2012)
12. Lin, C., He, Y.: Joint sentiment/topic model for sentiment analysis. In: CIKM, pp. 375–384 (2009)
13. Titov, I., McDonald, R.T.: A joint model of text and aspect ratings for sentiment summarization. In: ACL, pp. 308–316 (2008)
14. Jo, Y., Oh, A.H.: Aspect and sentiment unification model for online review analysis. In: WSDM, pp. 815–824 (2011)
15. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences of the United States of America 101(suppl. 1), 5228–5235 (2004)
16. Baccianella, S., Esuli, A., Sebastiani, F.: Multi-facet rating of product reviews. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) ECIR 2009. LNCS, vol. 5478, pp. 461–472. Springer, Heidelberg (2009)
17. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: ACL (2007)
18. Brody, S., Elhadad, N.: An unsupervised aspect-sentiment model for online reviews. In: HLT-NAACL, pp. 804–812 (2010)
19. Denecke, K.: Are sentiwordnet scores suited for multi-domain sentiment classification? In: ICDIM, pp. 33–38 (2009)

Fast Top- k Distance-Based Outlier Detection on Uncertain Data

Salman Ahmed Shaikh and Hiroyuki Kitagawa

Graduate School of Systems and Information Engineering

University of Tsukuba

1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan

salman@kde.cs.tsukuba.ac.jp, kitagawa@cs.tsukuba.ac.jp

Abstract. This paper studies the problem of top- k distance-based outlier detection on uncertain data. In this work, an uncertain object is modelled by a probability density function of a Gaussian distribution. We start with the Naive approach. We then introduce a populated-cell list (PC-list), a sorted list of non-empty cells of a grid (grid is used to index our data). Using PC-list, our top- k outlier detection algorithm needs to consider only a fraction of dataset objects and hence quickly identifies candidate objects for top- k outliers. An approximate top- k outlier detection algorithm is also presented to further increase the efficiency of our outlier detection algorithm. An extensive empirical study on synthetic and real datasets shows that our proposed approaches are efficient and scalable.

Keywords: Top- k Distance-based Outlier Detection, Uncertain Data, Gaussian Distribution, PC-list based Approach.

1 Introduction

Outlier detection is one of the most important data mining techniques with vital importance in many application domains including credit card fraud detection, network intrusion detection, environment monitoring, etc. Hawkins [4] defines an outlier as an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism.

Most of the earliest outlier detection techniques were given by statistics [6]. However, most statistical techniques are univariate, and in the majority of techniques, the parameter of distribution is difficult to determine. In order to overcome these problems several distance-based approaches for outlier detection have been proposed in data mining [5], [11], [14].

Due to the increasing usage of sensors, RFIDs and similar devices for data collection these days, data contains certain degree of inherent uncertainty. The causes of uncertainty may include limitation of equipments, absence of data and delay or loss of data in transfer. In order to get reliable results from such data, uncertainty needs to be considered in calculation. In this work we study the problem of top- k distance-based outlier detection on uncertain data following the Gaussian distribution.

In the following, uncertainty of data is modelled by the most commonly used PDF, i.e., the Gaussian distribution. Since the distance between uncertain data objects is very costly to compute, we introduce a populated-cell list (PC-list) based top- k outlier detection technique. PC-list is a sorted list of non-empty cells of a d -dimensional grid, where grid is used to index our data. Using PC-list, our top- k outlier detection algorithm needs to consider only a fraction of the dataset objects and hence quickly identifies candidate objects for top- k outliers. Furthermore an approximate top- k outlier detection algorithm is also presented to increase the efficiency of our outlier detection algorithm.

The rest of the paper is organized as follows. Sec. 2 surveys the related work. Sec. 3 formally defines the top- k distance-based outlier detection on uncertain datasets. The PC-list, the top- k algorithm and the approximate top- k algorithm are presented in Sec. 4. Sec. 5 contains an extensive experimental evaluation that demonstrates the efficiency and scalability of proposed techniques. Sec. 6 concludes our paper.

2 Related Work

Distance-based outliers detection approach was introduced by Knorr, et al. in [5]. They defined a point p to be an outlier if at most M points are within D -distance of p . They also presented a cell-based approach to efficiently compute the distance-based outliers. [9] formulated distance-based outliers as the top- t data points whose distance to their κ^{th} nearest neighbour is largest. Angiulli et al. in [10] gave a slightly different definition of outliers than [9] by considering the average distance to their k nearest neighbours. Besides, there are some works on the detection of distance-based outliers over stream data including [13], [14] and [15]. These works are based on the Knorr, et al. definition of distance-based outliers. Furthermore, [13] gave an approximate algorithm to reduce the memory space required by its exact counterpart. Later on [14] extended [13] work by adding the concepts of multi-query and micro-cluster based distance-based outlier detection. A geometric approach of outlier detection has also been proposed in [2]. The proposed solution is only suitable for identifying abnormal nodes from the cluster of nodes placed nearby and not valid for the problem when the measurements of a single node is classified as outliers, based on the nodes past measurements. However all these approaches were given for deterministic data and could not handle uncertain data.

Recently a lot of research has focused on managing, querying and mining of uncertain datasets [12], [7]. The problem of outlier detection on uncertain datasets was first studied by Aggarwal, et al. in [12]. They represented an uncertain object by a PDF. They defined an uncertain object o to be a density-based (δ, η) outlier, if the probability of o existing in some subspace of a region with density at least η is less than δ . However, their work focuses on detecting outliers in subspaces. In practise, an outlier in subspace is not necessarily an outlier in full space as argued in [11]. [7] also proposed a distance-based outlier detection algorithm on uncertain datasets, which was later extended in [8] for probabilistic data streams. However

in their works, an object's existential uncertainty is considered rather than representing an object by a PDF as in our work.

In [1], we proposed a cell-based approach of distance-based outlier detection on uncertain data. According to [1], an uncertain object o is a distance-based outlier if the expected number of objects lying within its D -distance is not greater than $M = N(1 - p)$, where N is the number of objects in the dataset and p is the fraction of objects that lies farther than D -distance of o . In practise parameter p is difficult to determine and is dependent on N . An arbitrary value of p may results in a very few or a lot of outliers for different N . Moreover from [1], we cannot obtain the outlier's ranking. Therefore in this work, we propose PC-list based approach of the top- k distance-based outlier detection, which can always obtain k strongest outliers along with their ranking, provided $k \leq N$.

3 Distance-Based Outliers in Uncertain Data

The very first definition of distance-based outlier detection on deterministic data was given by Knorr, et al. in [5]. They defined distance-based outliers as follows.

Definition 1. *An object o in a dataset DB is a distance-based outlier, if at least fraction p of the objects in DB lies greater than distance D from o .*

In this work, our focus is the detection of the top- k outliers on a dataset whose objects' attribute values are uncertain. This paper assumes that the uncertainty is given by the Gaussian distribution. The Gaussian distribution is chosen for representing uncertainty, because in statistics the Gaussian distribution (*or the normal distribution*) is the most important and the most commonly used.

In this paper, k -dimensional uncertain objects o_i are considered, with attribute $\vec{A}_i = (x_{i,1}, \dots, x_{i,k})^T$ following the Gaussian PDF with mean $\vec{\mu}_i = (\mu_{i,1}, \dots, \mu_{i,k})^T$ and co-variance matrix $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,k}^2)$, respectively. Namely, the vector \vec{A}_i is a random variable that follows the Gaussian distribution $\vec{A}_i \sim \mathcal{N}(\vec{\mu}_i, \Sigma_i)$. Note that $\vec{\mu}_i$ denotes the observed coordinates (attribute values) of object o_i . The complete database consists of a set of such objects, $\mathcal{GDB} = \{o_1, \dots, o_N\}$, where $N = |\mathcal{GDB}|$ is the number of uncertain objects in \mathcal{GDB} .

3.1 Top- k Distance-Based Outliers in Uncertain Data

We naturally extend Definition 1 for the top- k distance-based outliers on uncertain datasets as follows.

Definition 2. *The top- k distance-based outliers are the k uncertain objects in the dataset \mathcal{GDB} for which the expected number of objects lying within D -distance is smallest.*

The objects that lie within D -distance of an object o are called D -neighbours of o and the set of D -neighbours of o is denoted by $DN(o)$. In order to find the top- k

distance-based outliers in \mathcal{GDB} , the distance between uncertain objects needs to be calculated, which is given by another distribution known as the Gaussian difference distribution [3]. Let $\vec{\mathcal{A}}_i$ and $\vec{\mathcal{A}}_j$ be two independent d -dimensional normal random vectors with means $\vec{\mu_i} = (\mu_{i,1}, \dots, \mu_{i,d})^T$ and $\vec{\mu_j} = (\mu_{j,1}, \dots, \mu_{j,d})^T$ and diagonal covariance matrices $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,d}^2)$ and $\Sigma_j = \text{diag}(\sigma_{j,1}^2, \dots, \sigma_{j,d}^2)$, respectively. Then, $\vec{\mathcal{A}}_i - \vec{\mathcal{A}}_j = \mathcal{N}(\vec{\mu_i} - \vec{\mu_j}, \Sigma_i + \Sigma_j)$ [3]. Let $Pr(o_i, o_j, D)$ denotes the probability that $o_j \in DN(o_i)$. Then,

$$Pr(o_i, o_j, D) = \int_R \mathcal{N}(\vec{\mu_i} - \vec{\mu_j}, \Sigma_i + \Sigma_j) d\vec{\mathcal{A}}, \quad (1)$$

where R is a sphere with centre $(\vec{\mu_i} - \vec{\mu_j})$ and radius D . For the expression and derivation of $Pr(o_i, o_j, D)$, please refer our previous work [1]. Furthermore, we will use $Pr(\alpha, D)$ to denote $Pr(o_i, o_j, D)$ when there is no confusion, where α is an ordinary Euclidean distance between the means of $o_i \in \mathcal{GDB}$ and $o_j \in \mathcal{GDB}$. Computing this probability is usually very costly, and we have to avoid this computation as much as possible.

The Naive approach of the top- k outlier detection given in Alg. 1 uses Nested-loop. In order to find whether an object $o_i \in \mathcal{GDB}$ is a top- k outlier, we need to compute its expected D -neighbours ($EN(o_i)$). Computation of $EN(o_i)$ for an object $o_i \in \mathcal{GDB}$ requires evaluation of N expensive distance functions. During the computation of $EN(o_i)$, if expected D -neighbours become greater than threshold θ , o_i is an inlier and the computation of $EN(o_i)$ is stopped. On the other hand, if $EN(o_i)$ is less than or equal to θ , o_i is added to candidate list of outliers \mathbb{C}_{obj} , along with its expected D -neighbours. The \mathbb{C}_{obj} is kept sorted in ascending order of D -neighbours' column and the top- k objects in it are selected as outliers. In the worst case, this approach requires $O(N^2)$ evaluations of distance function, which is very expensive.

4 The Populated-Cells List (PC-list)

The Naive approach requires a lot of computation time to detect top- k outliers even from a small dataset due to the costly distance calculation. To overcome this problem we propose a PC-list-based approach of the top- k outlier detection. PC-list is an array of non-empty cells of a d -dimensional grid containing uncertain data objects $o \in \mathcal{GDB}$. The PC-list helps in detection of the top- k distance-based outliers by identifying the cells containing candidate outliers.

Lemma 1. *Let $o_i, o_j \in \mathcal{GDB}$ be two d -dimensional uncertain objects following the Gaussian distribution and α denotes an ordinary Euclidean distance between the means of o_i and o_j . Then for $t \in \mathcal{R}$, denoting the number of standard deviations required to enclose a large probability (say > 99%) of a d -dimensional Gaussian difference distribution, following statements hold.*

- (a) If $\alpha \leq D - t\sigma'$, $Pr(o_i, o_j, D) \approx 1$.
- (b) If $\alpha \geq D + t\sigma'$, $Pr(o_i, o_j, D) \approx 0$.

Algorithm 1. The top- k Naive Approach

Input: \mathcal{GDB}, D, k
Output: Top- k Distance-based Outliers

```

1:  $N \leftarrow |\mathcal{GDB}|, \theta \leftarrow \infty, \mathbb{C}_{obj} \leftarrow \phi$  (Candidate top- $k$  outliers list);
2: for each  $o_i$  in  $\mathcal{GDB}$  do
3:    $EN(o_i) \leftarrow 0$ ; (expected number of  $D$ -neighbours of  $o$ )
4:   for each  $o_j$  in  $\mathcal{GDB}$  do
5:      $EN(o_i) += Pr(o_i, o_j, D);$ 
6:     if  $EN(o_i) > \theta$  then GOTO next  $o_i$ ;
7:   end for
8:   Insert  $o_i$  and  $EN(o_i)$  into  $\mathbb{C}_{obj}$  (Keep  $\mathbb{C}_{obj}$  sorted w.r.t.  $EN(o)$ );
9:   if  $|\mathbb{C}_{obj}| > k$  then
10:    Set  $\theta = EN(o')$ , where  $o'$  is the  $k^{th}$  object in  $\mathbb{C}_{obj}$ ;
11:    Remove all  $o'' \in \mathbb{C}_{obj}$ , such that  $EN(o'') > \theta$ ;
12:   end if
13: end for
14: return  $\mathbb{C}_{obj};$ 
```

where σ' is the standard deviation of the Gaussian difference distribution in any one dimension (assuming that the standard deviation is uniform in all the dimensions).

Proof. The number of standard deviations s needed to enclose a given probability for a d -dimensional random variable X following the Gaussian distribution can be obtained using the expression $Pr\{d_M(X, \mu) \leq s\} = G_d(s^2)$ [19], where $d_M(X, \mu) = \sqrt{(X - \mu)^T \sum^{-1}(X - \mu)}$ is the Mahalanobis distance and $G_d(s^2)$ is the CDF of the chi-squared distribution with d -degrees of freedom.

Here we are interested in computing the distance between two uncertain objects o_i and o_j following the Gaussian distribution. This distance is given by another Gaussian distribution known as the Gaussian difference distribution [3]. Hence if t denotes the value of s , such that $Pr\{d_M(X, \mu) \leq t\}$ covers a large area of the Gaussian distribution (say $> 99\%$), then for $\alpha \leq D - t\sigma'$, $Pr(o_i, o_j, D) \approx 1$ and for $\alpha \geq D + t\sigma'$, $Pr(o_i, o_j, D) \approx 0$ ■

4.1 Structure

In order to find the top- k distance-based outliers from an uncertain dataset using the PC-list, we first quantize each object in \mathcal{GDB} , to a d -dimensional space that is partitioned into cells of length l (The cell length is discussed in Sec. 4.3). Let $C_{\psi_1, \dots, \psi_d}$ be any cell in grid \mathcal{G} , where positive integers ψ_1, \dots, ψ_d denote the cell indices. The layers (L_1, \dots, L_n) of $C_{\psi_1, \dots, \psi_d} \in \mathcal{G}$ are the neighbouring cells of $C_{\psi_1, \dots, \psi_d}$, as shown in Fig. 1 and are derived as follows.

$$\begin{aligned}
L_1(C_{\psi_1, \dots, \psi_d}) &= \{C_{x_1, \dots, x_d} | x_1 = \psi_1 \pm 1, \dots, x_d = \psi_d \pm 1, C_{x_1, \dots, x_d} \neq C_{\psi_1, \dots, \psi_d}\}. \\
L_2(C_{\psi_1, \dots, \psi_d}) &= \{C_{x_1, \dots, x_d} | x_1 = \psi_1 \pm 2, \dots, x_d = \psi_d \pm 2, \\
&\quad C_{x_1, \dots, x_d} \notin L_1(C_{\psi_1, \dots, \psi_d}), C_{x_1, \dots, x_d} \neq C_{\psi_1, \dots, \psi_d}\}.
\end{aligned}$$

$L_3(C_{\psi_1, \dots, \psi_d}), \dots, L_n(C_{\psi_1, \dots, \psi_d})$ are derived in a similar way. We will use C to denote $C_{\psi_1, \dots, \psi_d}$ when there is no confusion.

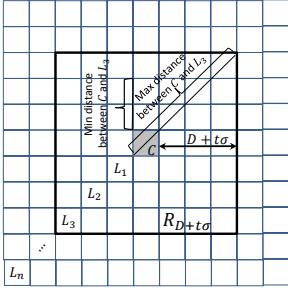


Fig. 1. Cell Layers and Bounds

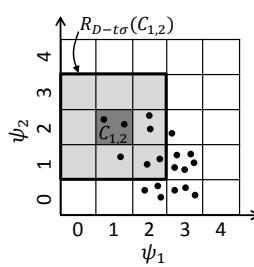


Fig. 2. PC-list building

Let $R_{D-t\sigma}(C)$ denotes a region formed by $\left\lfloor \frac{D-t\sigma}{l\sqrt{d}} - 1 \right\rfloor$ neighbouring layers of $C \in \mathcal{G}$. The region $R_{D-t\sigma}(C)$ is chosen in such a way that for each $o_i \in C$ and $o_j \in R_{D-t\sigma}(C)$, $Pr(o_i, o_j, D) \approx 1$. let $\mathcal{C}(C)$ is the count of objects in C , and $\mathcal{C}_{D-t\sigma}(C)$ is the count of objects within cells in region $R_{D-t\sigma}(C)$ (including C itself). Then the PC-list (PC) is a sorted list containing $\mathcal{C}(C)$ and $\mathcal{C}_{D-t\sigma}(C)$ for each non-empty cell $C \in \mathcal{G}$ as shown in Fig.2. The tuples in the PC-list are sorted in an ascending order of $\mathcal{C}_{D-t\sigma}(C)$ column. The idea behind sorting is that outliers tend to exist in sparse regions. Sorting tuples in the PC-list, lets us identify cells in sparse regions.

4.2 Cell Bounds

In order to identify cells $C \in PC$, containing only inliers or candidate top- k outliers, their bounds on the expected D -neighbours are used. A cell C can be pruned as an inlier cell if the minimum expected D -neighbours for any object in C is greater than threshold θ (θ is discussed shortly). Similarly a cell can be identified as containing top- k outliers if the maximum expected D -neighbours for any object in C is less than θ . Since the Gaussian distribution is unbounded, $Pr(o_i, o_j, D)$ is always greater than zero for $o_i, o_j \in \mathcal{G}$. Therefore all the cells in the PC-list need to be considered for the computation of bounds of $C \in PC$. To compute cell bounds, the minimum and the maximum ordinary Euclidean distances between cells are required. Beside distance between cells, object count of each $C \in PC$ and precomputed $Pr(\alpha, D)$ values for α ranging from the minimum to the maximum ordinary Euclidean distances between cells in \mathcal{G} are also required for the computation of $C \in PC$ bounds. The precomputed values are stored in a look-up table to be used by the top- k outlier detection algorithm.

Cell	$\mathcal{C}(C)$	$\mathcal{C}_{D-t\sigma}(C)$
C_{12}	2	7
C_{11}	1	10
C_{32}	1	10
C_{22}	2	13
C_{30}	3	13
C_{20}	3	14
C_{31}	5	16
C_{21}	2	19

Distance between Cells: Let C_p and C_q are two cells in PC with indices $\psi_{p1}, \dots, \psi_{pd}$ and $\psi_{q1}, \dots, \psi_{qd}$ respectively. Let $\Delta_{min}(C_p, C_q)$ and $\Delta_{max}(C_p, C_q)$ denote the minimum and the maximum ordinary Euclidean distances between C_p and C_q respectively. Distance between C_p and C_q depends on their positions in the grid \mathcal{G} and can be derived as follows.

$$\Delta_{min}(C_p, C_q) = l \cdot \left(\sum_{s=1}^d \delta_{min,s}^2 \right)^{1/2} \text{ where } \delta_{min,s} = \begin{cases} \psi_{ps} - (\psi_{qs} + 1) & \psi_{ps} > \psi_{qs} \\ (\psi_{ps} + 1) - \psi_{qs} & \psi_{ps} < \psi_{qs} \\ \psi_{ps} - \psi_{qs} & \psi_{ps} = \psi_{qs} \end{cases}$$

$$\Delta_{max}(C_p, C_q) = l \cdot \left(\sum_{s=1}^d \delta_{max,s}^2 \right)^{1/2} \text{ where } \delta_{max,s} = \begin{cases} (\psi_{ps} + 1) - \psi_{qs} & \psi_{ps} \geq \psi_{qs} \\ \psi_{ps} - (\psi_{qs} + 1) & \psi_{ps} < \psi_{qs} \end{cases}$$

Now we can obtain bounds for cells in the PC-list using pre-computed $Pr(\alpha, D)$ values and the information available in the PC-list. Let $LB(Pr(C_p, C_q))$ and $UB(Pr(C_p, C_q))$ denote $Pr(\alpha, D)$ values at minimum $\alpha \geq \Delta_{max}(C_p, C_q)$ and maximum $\alpha \leq \Delta_{min}(C_p, C_q)$ respectively. Then for a $C \in PC$, $LB(C) = (\sum_{C' \in PC} LB(Pr(C, C')) * \mathcal{C}(C'))$ and $UB(C) = (\sum_{C' \in PC} UB(Pr(C, C')) * \mathcal{C}(C'))$.

Let $R_{D+t\sigma}(C)$ denotes the region formed by $\lceil \frac{D+t\sigma}{l} \rceil$ neighbouring layers of cell $C \in \mathcal{G}$ as shown in Fig. 1. Region $R_{D+t\sigma}(C)$ is chosen in such a way that for each $o_i \in C$ and $o_j \notin R_{D+t\sigma}(C)$, $Pr(o_i, o_j, D)$ approaches zero. Since the major contribution in the bounds for $C \in \mathcal{G}$ is done by the cells in region $R_{D+t\sigma}(C)$, we redefine the bounds for $C \in PC$, to reduce the number of pre-computations and bounds computation time, as follows.

$$LB(C) = \left(\sum_{C' \in \{PC \cap R_{D+t\sigma}(C)\}} LB(Pr(C, C')) * \mathcal{C}(C') \right).$$

$$UB(C) = \left(\sum_{C' \in \{PC \cap R_{D+t\sigma}(C)\}} UB(Pr(C, C')) * \mathcal{C}(C') + \right.$$

$$\left. Pr(l\sqrt{d}(\lceil \frac{D+t\sigma}{l} \rceil + 1), D) * (N - \sum_{C' \in \{PC \cap R_{D+t\sigma}(C)\}} \mathcal{C}(C')) \right).$$

Number of Pre-computations: Since the bounds are pre-computed for the cells in region $R_{D+t\sigma}(C)$, $Pr(\alpha, D)$ values are computed only for the neighbouring layers within $D + t\sigma$ distance of a cell. For $\lceil \frac{D+t\sigma}{l} \rceil$ neighbouring layers, we require $2\lceil \frac{D+t\sigma}{l} \rceil$ pre-computations. Two more pre-computations are required for the cell C itself and the objects that lie greater than $D + t\sigma$ distance of a cell. Hence the total number of pre-computations required are only $2\lceil \frac{D+t\sigma}{l} \rceil + 2$.

4.3 Candidate Outlier Cells

Let \mathbb{C}_{cell} is a list containing candidate outlier cells from PC , sorted in ascending order of $UB(C)$. Let $C^k \in \mathbb{C}_{cell}$ is a cell with the minimum upper bound containing the k^{th} object. A $C \in PC$ is a candidate outlier cell whenever $\sum_{C' \in \mathbb{C}_{cell}} \mathcal{C}(C') < k$ or $LB(C) \leq \theta$, where $\theta = UB(C^k)$ denotes the threshold.

Cell Pruning and θ Updation: For a $C \in PC$, if $LB(C) > \theta$, C cannot contain any of the top- k outliers and can be pruned. On the other hand, if $LB(C) \leq \theta$, C may contain the top- k outlier. C is added to \mathbb{C}_{cell} , such that \mathbb{C}_{cell} remain sorted of its $UB(C)$ attribute. Set $\theta = UB(C^k)$ and remove C' from \mathbb{C}_{cell} , such that $LB(C') > \theta$, as they cannot contain the top- k outliers.

Stopping Condition: The PC-list is scanned from top to bottom for candidate outlier cells. During the scanning, if a $C' \in PC$ is found such that $Pr(D-t\sigma, D) * \mathcal{C}_{D-t\sigma}(C') > \theta$, neither C' nor any cell after it in PC-list can contain outliers. Hence the PC-list scanning can be stopped at this point.

Cell Length l : Due to the complexity of our distance function, it is not possible to derive a single cell length l suitable for all the combinations of D and variances. Very small cell length increases the number of cells in the Grid exponentially and the time required to construct the PC-list. A good starting point of the cell length that we found through experiments is the standard deviation, i.e., $l = \sigma$.

Algorithm 2. The Top- k Distance-based Outliers

Input: GDB, D, l, k

Output: Top- k Distance-based Outliers

```

1:  $N \leftarrow |\mathcal{GDB}|, \theta \leftarrow \infty;$ 
2:  $\mathbb{C}_{cell} \leftarrow \phi, \mathbb{C}_{obj} \leftarrow \phi;$  (Candidate outlier cells and top- $k$  outliers list respectively)
3: Create cell grid  $\mathcal{G}$  depending upon dataset values and cell length  $l$ ;
4: Map each  $o \in \mathcal{GDB}$  to an appropriate cell  $C \in \mathcal{G}$ ;
5: Create PC-list  $PC$ , using non-empty cells of  $\mathcal{G}$ ;
6: Sort  $PC$  w.r.t.  $\mathcal{C}_{D-t\sigma}(C)$  column;
   /*Searching candidate outlier cells*/
7: for each  $C$  in  $|PC|$  do
8:   if  $\mathcal{C}_{D-t\sigma}(C) * Pr(D - t\sigma, D)$  then Exit for loop. /*Stopping condition*/
9:   Compute  $LB(C)$  and  $UB(C)$ ;
10:  if  $LB(C) \leq \theta$  then
11:    Add  $C$  to  $\mathbb{C}_{cell}$  (keep  $\mathbb{C}_{cell}$  sorted of  $UB(C)$  attribute);
12:    if  $\mathbb{C}_{cell}$  contains  $\geq k$  objects then
13:      Set  $\theta = UB(C^k)$ , such that  $C^k$  contain the  $k^{th}$  object;
14:      Remove all  $C$  from  $\mathbb{C}_{cell}$ , such that  $LB(C) > \theta$ ;
15:    end if
16:  end if
17: end for
   /*Calculating  $EN(o)$  of candidate top- $k$  outliers*/

```

The computation of $EN(o)$ is similar to that of the Naive approach. The only difference is that in this algorithm we compute $EN(o)$ for the candidate objects in \mathbb{C}_{cell} only.

4.4 Outlier Detection Algorithms

In this section, we present two algorithms to detect top- k distance-based outliers from uncertain datasets. The first algorithm computes accurate expected

D -neighbours for all the un-pruned objects, however the second algorithm approximates the expected D -neighbours to reduce the algorithm computation cost.

Top- k Algorithm: The algorithm 2 first maps dataset objects to appropriate grid cells and creates the PC-list in lines 4 and 5 respectively. Since the PC-list is sorted in the ascending order of its $\mathcal{C}_{D+t\sigma}(C)$ column, it guarantees that cells in the sparse regions of the grid \mathcal{G} are at the top of the PC-list. Hence the candidate outlier cells are expected to be at the top of the list. We scan the PC-list and add the candidate outlier cells in \mathbb{C}_{cell} until the stopping condition on line 8 becomes true. The number of objects in \mathbb{C}_{Cell} may be greater than k , hence we calculate expected D -neighbours $EN(o)$ of candidate objects to find the top- k outliers and their ranking. The o is then added to the \mathbb{C}_{obj} (set of candidate outlier objects) along with its $EN(o)$. The objects in \mathbb{C}_{obj} are sorted in ascending order of $EN(o)$ column. As the k^{th} object's $EN(o)$ is found, threshold θ is set (refer line 10 of Alg.1). During the calculation of $EN(o)$, if for some o' , $EN(o')$ becomes greater than θ , then o' can not be among the top- k outliers and is removed from further consideration.

Approximate Top- k Algorithm: In the top- k algorithm, the minimum number of distance function computations required for the evaluation of k $EN(o)$ is kN , however the candidate outlier objects which require the evaluation of $EN(o)$, may be greater than k . When the distance function is expensive to compute (as in our case), computation of even k $EN(o)$ is very expensive. According to our distance function, the major contribution in the evaluation of $EN(o)$ is done by the nearer objects. Hence $EN(o)$ for each un-pruned o can be approximated with high accuracy by considering objects only within $D + t\sigma$ distance of o according to Lemma 1, rather than considering all the objects in dataset. Rest of the algorithm is same as that of Alg.2.

Maximum Approximation Error: For any $o \in \mathcal{GDB}$, maximum approximation error (ε_{max}) happens if all the $o' \in \mathcal{GDB} \setminus o$ are at a distance slightly greater than $D + t\sigma$ from o . Hence $\varepsilon_{max} = (N - 1) * Pr(D + t\sigma + \beta, D)$, where $\beta \in \mathbb{R}$ is a very small real value to make distance greater than $D + t\sigma$.

For example for $t = 9$, $d = 2$ and $N = 10^5$ objects, $\varepsilon_{max} \approx 10^{-5}$. ε_{max} depends mainly on t . In practice $t \geq 6$ gives sufficiently accurate $EN(o)$ for $d = 2$ and 3. For higher d values, we need to increase t value according to Lemma 1.

5 Experiments

We conducted extensive experiments on synthetic and real data to evaluate the effectiveness and scalability of our proposed algorithms. All algorithms were implemented in C++, GNU compiler. All experiments were performed on a system with an Intel Core 2 Duo CPU E8400 3.00GHz CPU and 2GB main memory running Ubuntu 12.04 OS. All programs run in main memory and no I/O cost is considered.

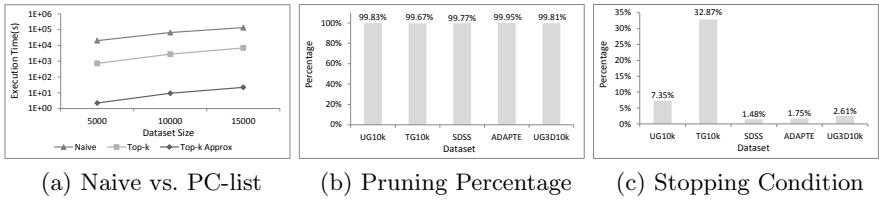


Fig. 3. Effectiveness of the PC-list based approach

We use two synthetic datasets and two real datasets for our experiments. Synthetic datasets, unimodal Gaussian (UG) and trimodal Gaussian (TG) are 2-dimensional and are generated using BoxMuller method [16]. A 3-dimensional unimodal Gaussian (UG3D) dataset is also generated for the evaluation of our proposed approaches on 3-dimensional data. A shorthand notation “*DatasetName + DatasetSize*” (e.g. UG5k to denote 5,000 tuples of unimodal Gaussian dataset) is used in figures. As for real-world data, we use two datasets: ADAPTE and SDSS. ADAPTE is obtained from CISL Research data archive [17] and SDSS is obtained from Sloan Digital Sky Survey [18]. ADAPTE consists of about 1,851 maximum and minimum temperature values collected from the National Polytechnic Institute of Mexico and National Meteorological System. SDSS dataset contains 10,136 Right Ascension (or ”RA”) and Declination (or ”Dec”) coordinates of stars and galaxies.

All the datasets are normalized to have a domain of [0,1000] on every dimension. For each point z in a dataset, we create an uncertain object o , whose uncertainty is given by Gaussian distribution with mean z and standard deviation σ in all the dimensions. Unless specified, following parameter values are used: $D = 100$, $\sigma = 10$, $l = 10$ and $k = 0.1\%$ of the respective dataset size. For approximate top- k algorithm, we considered objects only within $D + 6\sigma$ distance of each un-pruned object o . Pre-computation time is not included in the measurements. We first conduct experiments to evaluate the efficiency of our proposed algorithms presented in Sec.4.4. Fig. 3a compares the execution times of Naive and proposed algorithms on UG dataset. Our proposed algorithms are several times faster than its Naive counterpart due to their strong pruning capability as can be observed from Fig.3b. Stopping condition discussed in Sec.4.3 helps identify candidate outlier cells very quickly. Fig.3c shows the percentage of cells considered in the PC-list to identify candidate outlier cells. The percentage is comparatively higher for trimodal Gaussian dataset because the dataset is relatively sparse and hence results in larger number of candidate outlier cells. Moreover the approximate top- k algorithm is thousands of times faster than the ordinary top- k algorithm, due to the reason discussed in Sec.4.4. From theoretical analysis in Sec. 4.4 and experiments we found that the approximate top- k algorithm gives an accuracy of up to several decimal digits in the evaluation of $EN(o)$ and hence the outliers obtained from both the algorithms are same. Therefore in the rest of this section, we will evaluate only approximate top- k algorithm.

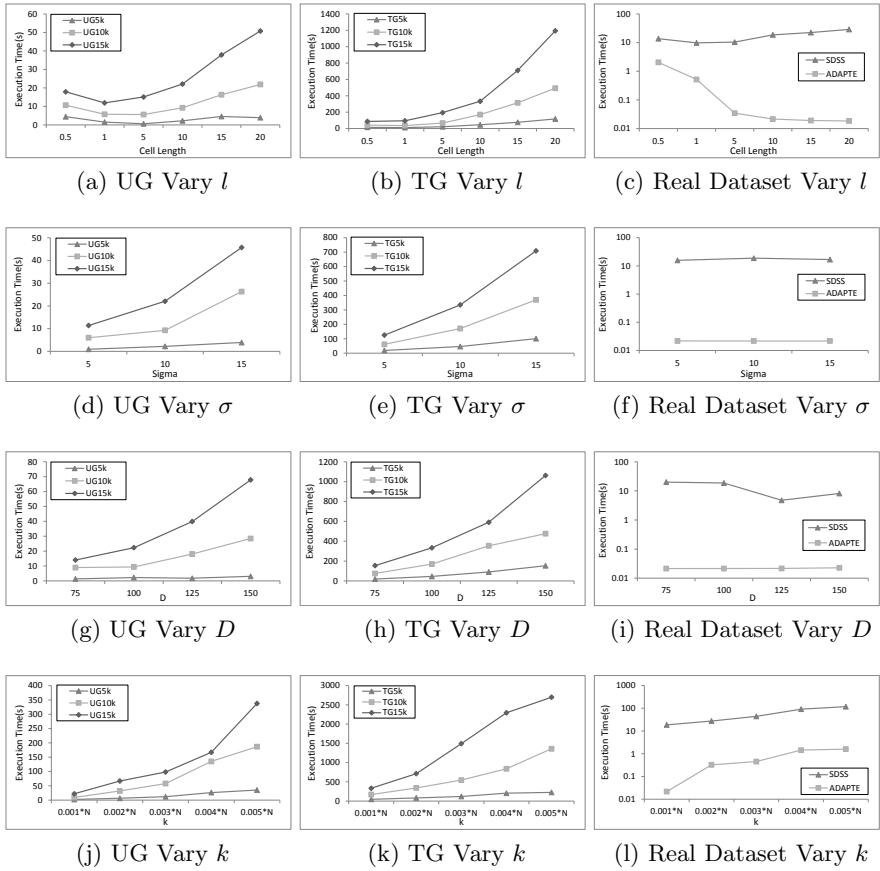


Fig. 4. Varying parameters l , σ , D and k for 2D datasets

Graphs in Fig.4 show the effect of varying different parameters on the execution times. It is obvious from the graphs in Figs. 4a, 4b and 4c that smaller cell lengths require lower execution times. However very small cell length increases the number of cells exponentially and therefore the execution time of the algorithm. Therefore we recommend the use of cell length equal to the standard deviation as discussed in Sec. 4.3. In Fig. 4c, k is very small due to the small size of ADAPTE dataset and therefore pruning time dominates the algorithm execution time. Consequently as the number of cells decreases due to the increase in cell length, algorithm execution time decreases. Next we perform experiments by varying the parameter σ . As σ increases, the uncertainty of the object also increases. This increase in uncertainty results in smaller $Pr(o_i, o_j, D)$ values even if o_i and o_j are located nearby. Hence the number of distance function evaluations required increases for un-pruned objects, which results in higher execution times as can be observed from Figs. 4d, 4e and 4f. Figs. 4g, 4h and 4i show the effect of varying parameter D . For each un-pruned o from the PC-list-based pruning,

increase in D results in an increase in the D -neighbours which need to be considered for the approximation of $EN(o)$. Therefore it increases the execution time of the overall algorithm for larger values of D . From Figs. 4j, 4k and 4l, increase in k results in an increase in execution time of algorithm, which is quite obvious behaviour of our algorithm. Figure 5 shows similar effects of varying different parameters on three dimensional dataset.

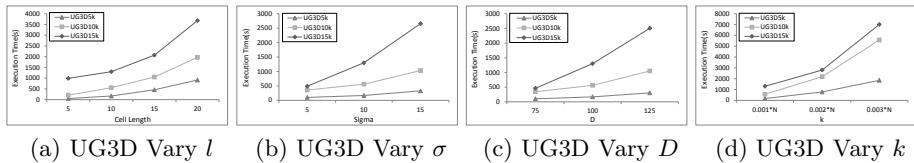


Fig. 5. Varying parameters l , σ , D and k for 3D dataset

6 Conclusion

In this work, the top- k distance-based outlier detection approach on uncertain datasets of the Gaussian distribution based on the PC-list is proposed. PC-list helps identify candidate outlier objects very quickly without considering all the objects in dataset. Moreover an approximate top- k outlier detection approach is presented to further reduce the algorithm computation cost. An extensive empirical study on real and synthetic datasets demonstrate the effectiveness and scalability of our proposed approaches.

Acknowledgement: This work has been partly supported by Grant-in-Aid for Scientific Research(A)(#24240015A).

References

1. Shaikh, S.A., Kitagawa, H.: Distance-Based Outlier Detection on Uncertain Data of Gaussian Distribution. In: Sheng, Q.Z., Wang, G., Jensen, C.S., Xu, G. (eds.) APWeb 2012. LNCS, vol. 7235, pp. 109–121. Springer, Heidelberg (2012)
2. Burdakis, S., Deligiannakis, A.: Detecting Outliers in Sensor Networks Using the Geometric Approach. In: ICDE (2012)
3. Weisstein, E.W.: Normal Difference Distribution, From MathWorld - A Wolfram Web Resource, <http://mathworld.wolfram.com>
4. Hawkins, D.: Identification of Outliers. Chapman and Hall (1980)
5. Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-Based Outliers: Algorithms and Applications. The VLDB Journal (2000)
6. Barnett, V., Lewis, T.: Outliers in Statistical Data. John Wiley (1994)
7. Wang, B., Xiao, G., Yu, H., Yang, X.: Distance-Based Outlier Detection on Uncertain Data. In: ICCIT (2009)
8. Wang, B., Yang, X., Wang, G., Yu, G.: Outlier detection over sliding windows for probabilistic data streams. Journal of Comp. Sc. & Tech. 25(3) (2010)

9. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient Algorithms for Mining Outliers from Large Data Sets. In: ACM, SIGMOD (2000)
10. Angiulli, F., Pizzuti, C.: Fast Outlier Detection in High Dimensional Spaces. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 15–27. Springer, Heidelberg (2002)
11. Nguyen, H.V., Gopalkrishnan, V., Assent, I.: An unbiased distance-based outlier detection approach for high-dimensional data. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 138–152. Springer, Heidelberg (2011)
12. Aggarwal, C.C., Yu, P.S.: Outlier Detection with Uncertain Data. In: SDM (2008)
13. Angiulli, F., Fassetti, F.: Detecting distance-based outliers in streams of data. In: CIKM (2007)
14. Kontaki, M., Gounaris, A., Papadopoulos, A.N., Tsichlas, K., Manolopoulos, Y.: Continuous monitoring of distance-based outliers over data streams. In: ICDE (2011)
15. Ishida, K., Kitagawa, H.: Detecting Current Outliers: Continuous Outlier Detection over Time-Series Data Streams. In: Bhownick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 255–268. Springer, Heidelberg (2008)
16. Thistleton, W., Marsh, J.A., Nelson, K., Tsallis, C.: Generalized Box-Muller method for generating q-Gaussian random deviates. IEEE Trans. on Info. Theory (2007)
17. CISL Research Data Archive, <http://rda.ucar.edu>
18. Sloan Digital Sky Survey, <http://www.sdss.org>
19. Bajorski, P.: Statistics for Imaging, Optics and Photonics. A John Wiley & Sons Publication (2012)

A Metric Learning Based Approach to Evaluate Task-Specific Time Series Similarity

Yang Lu, Wayne Xin Zhao, Hongfei Yan*, and Xiaoming Li

Peking University, Beijing, China

{andimeoltj, batmanfly, yhf1029}@gmail.com, lxm@pku.edu.cn

Abstract. A variety of methods have been proposed to measure time series similarity, such as Dynamic Time Warping and Edit distance. Although these methods have been shown to be effective and useful in various data mining tasks, they seldom consider task-specific information. Without consideration of task-specific information, the similarity measures may not work quite well on specific tasks. In this paper, we investigate how to learn task-specific time series similarity measures. We adopt metric learning as the principled approach, and we proposed two novel models based on metric learning to evaluate task-specified time series similarity. We construct our test collection based on real data from Renren Games data. Extensive experimental results show that our proposed methods are very effective.

1 Introduction

Evaluating time series similarity has been an important research problem and plays a very crucial role in many data mining algorithms. There is a considerable amount of methods for measuring the similarity between time series. Generally speaking, these methods evaluate the similarity based on surface patterns of time series: two time series are more similar if they share more similar surface patterns. For example, DTW [1] and LCSS [2] applies dynamic programming technique to capture common patterns which allows for time shifting between the two time series. These methods are widely used and serve as a foundation in various data mining tasks such as [3,4].

However, these methods are usually developed in a general setting but not based on the specific tasks. Take time series classification as one example, the major aim is to identify the discriminative patterns between inter-class time series. A good similarity evaluating method should not only capture large intra-class similarity of time series but also discriminate inter-class time series. Many applications have their own specific focus and characteristics, therefore general methods for evaluating time series similarity may not work well in specific tasks.

To address this problem, a promising research direction is to develop task-specific similarity evaluation methods for time series. That is the research topic we focus on in this paper. Specifically, we take time series classification as the major task and study a real task: user churn prediction for online games. We take real data of a very large Chinese online game platform, i.e., Renren Games¹. For each user, we can observe multiple time series each of which corresponds to the records of a feature within a

* Corresponding author.

¹ <http://wan.renren.com>

given time span, e.g. daily online playing time. Our problem can be summarized as follows: given a time series consisting of log information from a user, we would like to judge whether it is from a normal user or an abnormal user who will churn from current game later.

To model task-specific time series similarity, we turn to a widely adopted technique in machine learning community, i.e. metric learning. Metric learning aims to learn a linear transformation of the original feature space (or learn a Mahalanobis distance metric) based on the specific objective function which is derived from specific tasks. Metric learning methods can be formulated in either a supervised (or semi-supervised) or an unsupervised way. In a supervised way, most metric learning methods attempt to learn a distance metric from side information which is often available in the form of pairwise constraints, that is, pairs of similar data points and pairs of dissimilar data points [5]. The information of similarity or dissimilarity between a pair of examples can be easily collected from the labeled information in supervised classification. We follow the supervised approach of metric learning and try to incorporate task-specific labeled information. With labeled information, we expect the derived metric can better adapt to our churn prediction task.

In this paper, we propose two novel methods based on metric learning to evaluate task-specific time series similarity. The first model is formulated to capture global similarity constraints (i.e. must-links between all the instance pairs within the same class), and global dissimilarity constraints (i.e. cannot-links between all the instance pairs from different classes). Compared to the first model, the seconde model is formulated in a local way. For each data instance, we build homogeneous and heterogeneous neighborhood and only model constraints between it and the other instances from its neighborhood. We construct our evaluation set based on real Renren Online Game data and we select several state-of-the-art time series similarity evaluation methods as baselines. Extensive experimental results show that our proposed methods are very effective.

2 Related Work

Time Series Similarity Evaluation. There is a large body of existing techniques for measuring the similarity between different time series. The Euclidean measure sums the Euclidean distance between points in each time series. Recently, dynamic programming based measures are shown to be both effective and efficient, include Dynamic Time Warping(DTW) [1], Edit distance with Real Penalty(ERP) [6], the Longest Common Subsequence(LCSS) [2], and Edit Distance on Real sequences(EDR) [3]. Another line is to use dimension reduction techniques, such as Singular value decomposition (SVD) [7] and Discrete Wavelet Transform [8]. They are closely related to our methods since the essence of our methods is dimension reduction, too. To the best of our knowledge, these methods usually work in an unsupervised way and it is difficult to incorporate useful task information.

Time Series Classification. Time series classification has been an important topic and attracted much attention from research communities [9,10,4,11]. Although many algorithms have been proposed, it has been shown [10] that k -nearest-neighbor (k -NN), especially when $k = 1$, with Euclidean distance is a very robust baseline and difficult to beat. Our major focus is not to propose new methods for time series classification but examine whether the task-specific similarity can improve the classification performance

or not. Therefore, we follow the method presented in [10], using k -NN as our classifier. By combining the classifier with different similarity measures, we can compare the classification performance and check which measure performs best.

Metric Learning. Our methods are highly built on the related research of metric learning [12,5,13]. The most prominent form of metric learning is to learn the Mahalanobis metric. Its computation can be seen as a two-step process: in the first step we perform a linear projection of the instances and in the second step we compute their Euclidean distance in the projected space. Since many data mining algorithms rely on the underlying similarity measure, metric learning provides a principled approach to automatically derive a reasonable measure according to specific tasks. In this paper, our focus is how to adapt existing metric learning methods to evaluate task-specific time series similarity.

3 Learning Task-Specific Time Series Similarity

We focus on the task of time series classification. In this section, we propose two metric learning algorithms to evaluate task-specific time series similarity².

First of all, let us introduce some preliminary notations. Let $\mathbf{s} = \{\bar{s}_1, \dots, \bar{s}_L\}$ denote a *time series* of length L represented as a column vector, where each value \bar{s}_i is a real number on the i th timestamp. Let \mathcal{Y} denote the label set. Without loss of generality, we consider the two-class classification problem, i.e. $\mathcal{Y} = \{+1, -1\}$, where "+1" denotes the positive class and "-1" denotes the negative class. Let $\mathcal{S}_{train} = \{(\mathbf{s}_i, y_i)\}_{i=1}^N$ denote a training set of N labeled examples (or data instances) with a time series of \mathbf{s}_i as input and corresponding class label $y_i \in \mathcal{Y}$. Let $\mathcal{S}_{test} = \{(\mathbf{s}_i)\}_{i=1}^M$ denote a test set of M unlabeled examples for which we would like to predict the class labels. We further assume that examples in training set and test set are of the same length L .

The essence of metric learning is to learn a *Mahalanobis* distance which is defined as follows

$$d_{\Sigma}(\mathbf{s}_i, \mathbf{s}_j) = \sqrt{(\mathbf{s}_i - \mathbf{s}_j)^T \Sigma (\mathbf{s}_i - \mathbf{s}_j)}, \quad (1)$$

where Σ is defined to be a symmetric positive semi-definite matrix of size $L \times L$. Since Σ is a symmetric positive semi-definite matrix, we can find a non-square matrix \mathbf{W} of size $L \times l$, with $l \leq L$, such that $\Sigma = \mathbf{W}\mathbf{W}^T$. Then the *Mahalanobis* distance between \mathbf{s}_i and \mathbf{s}_j induced by Σ , i.e. $d_{\Sigma}(\mathbf{s}_i, \mathbf{s}_j)$, becomes $\sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}$, where $\mathbf{x}_i = \mathbf{W}^T \mathbf{s}_i$ is a transformed time series of length l . Thus \mathbf{W} actually defines a mapping from the original input space to a lower dimension space. In this way, learning an optimal distance metric is equivalent to finding an optimal projection matrix. We expect the similarity derived on the projection space can help improve the performance of time series classification.

In the following, we present the proposed methods for evaluating task-specific time series similarity. The main idea is to construct two sets of constraints based on the specific task, and incorporate them into the metric learning framework. The first method is to consider these constraints on the entire data set while the second method is to consider these constraints only based on each example's neighborhood.

² We will use "similarity" and "distance" alternatively and do not explicitly discriminate between them, since given a similarity value we can easily transform it into a corresponding distance value.

3.1 Global Metric Learning

With the labels attached to each time series, we can define two sets of constraints called as *must-link* and *cannot-link* respectively, denoted by \mathcal{M} and \mathcal{C} :

$$\begin{aligned}\mathcal{M} : \forall (\mathbf{s}_i, \mathbf{s}_j) \in \mathcal{M} & , \quad \mathbf{s}_i, \mathbf{s}_j \text{ have the same label;} \\ \mathcal{C} : \forall (\mathbf{s}_i, \mathbf{s}_j) \in \mathcal{C} & , \quad \mathbf{s}_i, \mathbf{s}_j \text{ have different labels.}\end{aligned}$$

Our main idea is to maximize the average distance between inter-class examples but minimize the average distance between intra-class examples. We define the objective function as follows

$$\delta = \sum_{(\mathbf{s}_i, \mathbf{s}_j) \in \mathcal{C}} \frac{d_{\Sigma}^2(\mathbf{s}_i, \mathbf{s}_j)}{N_C} - \sum_{(\mathbf{s}_a, \mathbf{s}_b) \in \mathcal{M}} \frac{d_{\Sigma}^2(\mathbf{s}_a, \mathbf{s}_b)}{N_M}, \quad (2)$$

where $d_{\Sigma}(\cdot, \cdot)$ is the Mahalanobis distance between two time series defined in Eq. 1, N_M and N_C represent the numbers of *must-links* and *cannot-links* respectively. We can see the above equation tries to seek a trade-off between a small average intra-class distance and a large average inter-class distance.

By substituting Σ with $\mathbf{W}\mathbf{W}^T$, we can expand Eq. 2 as follows

$$\delta = \sum_{(\mathbf{s}_i, \mathbf{s}_j) \in \mathcal{C}} \frac{(\mathbf{s}_i - \mathbf{s}_j)^T \mathbf{W} \mathbf{W}^T (\mathbf{s}_i - \mathbf{s}_j)}{N_C} - \sum_{(\mathbf{s}_a, \mathbf{s}_b) \in \mathcal{M}} \frac{(\mathbf{s}_a - \mathbf{s}_b)^T \mathbf{W} \mathbf{W}^T (\mathbf{s}_a - \mathbf{s}_b)}{N_M}.$$

With some derivations, we can further rewrite δ with the help of *trace* operator in linear algebra

$$\delta = 2\text{tr}(\mathbf{W}^T(\mathbf{C} - \mathbf{M})\mathbf{W}),$$

where $\text{tr}(\cdot)$ denotes the *trace* of a matrix, \mathbf{C} and \mathbf{M} are constraint matrices defined as

$$\begin{aligned}\mathbf{C} &= \sum_{(\mathbf{s}_i, \mathbf{s}_j) \in \mathcal{C}} \frac{(\mathbf{s}_i - \mathbf{s}_j)(\mathbf{s}_i - \mathbf{s}_j)^T}{N_C}, \\ \mathbf{M} &= \sum_{(\mathbf{s}_a, \mathbf{s}_b) \in \mathcal{M}} \frac{(\mathbf{s}_a - \mathbf{s}_b)(\mathbf{s}_a - \mathbf{s}_b)^T}{N_M}.\end{aligned}$$

Based on the above discussion, we can formulate our global metric learning algorithm as

$$\begin{aligned}\mathbf{W} &= \arg \max_{\mathbf{W}} \text{tr}(\mathbf{W}^T(\mathbf{C} - \mathbf{M})\mathbf{W}), \\ s.t. \quad \mathbf{W}^T \mathbf{W} &= I.\end{aligned} \quad (3)$$

where the constraint $\mathbf{W}^T \mathbf{W} = I$ is to restrict the scale of \mathbf{W} . The solution to Eq. 3 is the largest l eigenvectors of $\mathbf{C} - \mathbf{M}$ and the optimal value of the objective function is the sum of the largest l eigenvalues of $(\mathbf{C} - \mathbf{M})$ [5,14]. We denote this method as **GMLS** (Global Metric Learning for evaluating time series Similarity).

3.2 Local Metric Learning

In the above method, given an example, we have to consider the distance constraint between it and all the other examples in the training set, either intra-class distance or inter-class distance. The assumption is a bit too restrictive, indeed, an example is not necessarily similar to all the other examples which have the same label. To relax the assumption, we further propose a local metric learning method for evaluating time series similarity, denoted as **LMLS**.

We still use previous *must-link* and *cannot-link* constraint sets, but only consider these constraints between examples in a local neighborhood. First, we propose two kinds of neighborhood, namely *cannot-link neighborhood* and *must-link neighborhood*. We use \mathcal{N}_i^C to denote the i^{th} time series' cannot-link neighborhood which consists of k nearest time series of s_i with different labels. And we use \mathcal{N}_i^M to denote the i^{th} time series' must-link neighborhood which consists of k nearest time series with the same label. We can apply previous time series similarity measures to build the neighborhood of a time series, in this paper, we use Euclidean distance due to its simplicity and efficiency. With these definitions, we can define the discrimination criterion for the i^{th} time series as

$$\delta_i = \sum_{s_a \in \mathcal{N}_i^C} d_{\Sigma}^2(s_i, s_a) - \sum_{s_b \in \mathcal{N}_i^M} d_{\Sigma}^2(s_i, s_b),$$

which tries to make time series with the same label more compact and those with different labels more diverse in a local neighborhood.

By going through all the examples in the training set, we can learn an optimal distance metric by maximizing the following objective function

$$\delta = \sum_{i=1}^N \delta_i.$$

Similar to GMLS, with the substitution of $\Sigma = \mathbf{W}^T \mathbf{W}$ and the application of *trace* trick, we can rewrite our objective function as

$$\delta = \text{tr}(\mathbf{W}^T (\mathbf{C}_{\mathcal{N}} - \mathbf{M}_{\mathcal{N}}) \mathbf{W}),$$

where $\text{tr}(\cdot)$ is the matrix trace and $\mathbf{C}_{\mathcal{N}}$ and $\mathbf{M}_{\mathcal{N}}$ are neighborhood based constraint matrices defined as

$$\begin{aligned} \mathbf{C}_{\mathcal{N}} &= \sum_i \sum_{s_a \in \mathcal{N}_i^C} (s_i - s_a)(s_i - s_a)^T, \\ \mathbf{M}_{\mathcal{N}} &= \sum_i \sum_{s_b \in \mathcal{N}_i^M} (s_i - s_b)(s_i - s_b)^T. \end{aligned}$$

Based on the above discussion, the distance metric learning problem can be formulated as

$$\begin{aligned} \mathbf{W} &= \arg \max_{\mathbf{W}} \text{tr}(\mathbf{W}^T (\mathbf{C}_{\mathcal{N}} - \mathbf{M}_{\mathcal{N}}) \mathbf{W}), \\ \text{s.t. } \mathbf{W}^T \mathbf{W} &= \mathbf{I}. \end{aligned} \tag{4}$$

Table 1. Basic explanations of six features. Note that all the log information of these features are accurate to day. For more details about these attributes, please refer to the offical website of “Luan Shi Tian Xia”.

Notations	Explanations
group	# of groups that a user is engaged in.
general	# of generals (i.e. a character in the game) a user trained.
completed	The percentage of game process a user completed.
money	The money a user consumed in the game.
logCount	The count a user logged in the game.
playTime	The total time a user spent in the game.

The solution of \mathbf{W} is the biggest l eigenvectors of $(\mathbf{C}_{\mathcal{N}} - \mathbf{M}_{\mathcal{N}})$.

4 Experiment

In this section, we present experimental setup and results.

4.1 Experimental setup

Construction of the Test Collection. We construct the test collection using the real data from “Luan Shi Tian Xia”, a popular online game issued by RenRen Games³. We study an important task in online gaming service: churn prediction of users. In this task, a label “+1” indicates “normal user” while a label “-1” indicates “churn users”. Given a user, the server keeps her log information of various features within a time span, e.g., per day’s playing time, which can be represented as a time series. With a time series as input, our task is to predict whether it is from an normal user or a churn user. Therefore, the task of churn prediction becomes time series classification.

In this paper, we consider a two-month time span: from June 1st, 2012 to July 31st, 2012. A user is considered to be active if he has logged to play this game at least 40 days in this time span. We only focus on active users and extract their log information of various features. Then we examine the log of August, 2012: if a user did not log to play game anymore, she is labeled as a churn user; otherwise, she is labeled as an normal user. Finally, we randomly select 500 normal users and 492 churn users.

By discussing with domain experts from Renren Games, we select six major features which are potentially important indicators to user churn behaviors. We summarize basic explanations of these features in Table 1. So for each user, we can generate six time series corresponding to daily activity statistics of these six features. Note that all the log information is accurate to day, and all the time series over these two months are of length 60. We scaled each time series by dividing the its largest numeric value. We totally have $992 \times 6 = 5952$ time series with labels and we further group these time series by features. For each feature, we randomly partition the data set into five folds, four folds are used as the *training set*, and one fold is used as the *test set*.

Methods to Compare. In this paper, our major focus is to examine whether task-specific similarity measure can beat general similarity measures for time series classification. We adopt k -NN as the classifier and compare the performance with different

³ <http://lstx.renren.com>

measures. We consider a few widely used methods to evaluate time series similarity. The first baseline measure is the Euclidean measure (EUC). Then we consider dynamic programming based methods include Dynamic Time Warping (DTW) [1], Edit distance with Real Penalty (ERP) [6], the Longest Common Subsequence (LCSS) [2], and Edit Distance on Real sequences (EDR) [3]. We select these methods since they are shown to be effective and efficient. Among these five baselines, Euclidean distance and ERP are metric, and they obey triangle inequality, which is an important property for indexing [3]. Note that our distance learning method naturally leads to measures which are metric⁴. For non-metric measures, it has been shown that EDR is more robust to noise than DTW and LCSS [3], in this paper, we also have similar findings, so we only report the results of EUC, ERP and EDR due to the space limit. Besides these baselines, we also consider a few dimension reduction methods for time series similarity as comparisons: Singular value decomposition (SVD) [15] and Discrete Wavelet Transform (DWT) [8]. These two methods are unsupervised dimension reduction methods. SVD seeks a low-dimension representation for original time series, to achieve this, we use all the data for SVD. These similarity measures have some parameters to tune. We use a grid search method to find the parameter setting which leads to the optimal average performance using five-fold cross-validation.

Evaluation Metrics. We use the following widely adopted measures for classification: *Recall*, *Precision* and *F1 score*. For each feature, we compute the values of these three metrics and only report the results of F1, since using the other two metrics lead to similar findings. Finally, we further average F1 scores of these six features.

4.2 Experimental Results

Performance Comparison. In Table 2, we present the results of k -NN classifier with different similarity measures. To check the stability of different measures, we vary the neighborhood size k in k -NN. We can have the following observations:

- 1) The value of k affects the performance of all the similarity (distance) measures. $k = 10$ gives the best performance and the results are relatively stable when $k > 10$.
- 2) Among all the baselines, EUC performs worst. EDR and SVD overall work better than others. The reason why SVD performs well may be we perform it on the entire data set, including training set and test set.
- 3) LMLS performs best in terms of average F1 scores. By zooming into the detailed results for single feature, we can see that LMLS performs best for the first three features while GLMS performs best for the last two features. Note that the major difference between LMLS and GLMS is that LMLS only considers distance constraints between an example of training set and corresponding examples in its neighborhood while GLMS considers distance constraints in the entire training set. It indicates that for some features we need large neighborhood to capture correlation between examples where GLMS is more suitable, but overall a localized neighborhood is more effective since it only considers the distance constraints between very similar examples in the original space.
- 4) All the methods achieve their optimal performance on the feature *group*, i.e., the number of groups that a user is engaged in the game. We have tried merging the time series of six features into a long time series, and then performed the classification task

⁴ [http://en.wikipedia.org/wiki/Metric_\(mathematics\)](http://en.wikipedia.org/wiki/Metric_(mathematics))

Table 2. Comparison of individual feature's F1 scores and average F1 score

	Methods	completed	logTime	logCount	group	general	money	Average
K=1	EUC+knn	0.602	0.594	0.618	0.814	0.475	0.547	0.608
	ERP+knn	0.606	0.614	0.658	0.834	0.554	0.574	0.640
	EDR+knn	0.576	0.669	0.644	0.808	0.540	0.580	0.636
	SVD+knn	0.631	0.574	0.634	0.783	0.574	0.628	0.638
	DWT+knn	0.590	0.571	0.665	0.838	0.594	0.584	0.640
	GMLS+knn	0.600	0.635	0.560	0.733	0.599	0.658	0.631
	LMLS+knn	0.633	0.708	0.717	0.758	0.589	0.559	0.661
K=3	EUC+knn	0.622	0.611	0.659	0.786	0.435	0.523	0.606
	ERP+knn	0.587	0.653	0.649	0.854	0.544	0.603	0.649
	EDR+knn	0.532	0.708	0.716	0.877	0.556	0.622	0.668
	SVD+knn	0.647	0.637	0.686	0.830	0.584	0.669	0.676
	DWT+knn	0.617	0.616	0.657	0.844	0.584	0.589	0.651
	GMLS+knn	0.630	0.661	0.534	0.749	0.599	0.683	0.643
	LMLS+knn	0.684	0.742	0.717	0.825	0.594	0.564	0.688
K=5	EUC+knn	0.621	0.648	0.689	0.777	0.449	0.573	0.626
	ERP+knn	0.579	0.660	0.680	0.861	0.570	0.599	0.658
	EDR+knn	0.567	0.754	0.726	0.870	0.571	0.660	0.691
	SVD+knn	0.683	0.647	0.692	0.838	0.604	0.614	0.680
	DWT+knn	0.604	0.639	0.663	0.839	0.581	0.618	0.657
	GMLS+knn	0.611	0.697	0.571	0.809	0.646	0.678	0.669
	LMLS+knn	0.713	0.777	0.702	0.824	0.595	0.574	0.697
K=10	EUC+knn	0.615	0.689	0.698	0.767	0.492	0.572	0.639
	ERP+knn	0.622	0.675	0.697	0.845	0.571	0.648	0.677
	EDR+knn	0.602	0.728	0.748	0.862	0.564	0.681	0.697
	SVD+knn	0.701	0.704	0.697	0.852	0.585	0.624	0.694
	DWT+knn	0.658	0.682	0.673	0.847	0.622	0.603	0.681
	GMLS+knn	0.605	0.709	0.655	0.806	0.639	0.679	0.682
	LMLS+knn	0.708	0.816	0.717	0.815	0.623	0.639	0.720
K=15	EUC+knn	0.585	0.737	0.679	0.784	0.477	0.544	0.634
	ERP+knn	0.614	0.702	0.664	0.828	0.652	0.610	0.678
	EDR+knn	0.605	0.743	0.741	0.857	0.560	0.687	0.699
	SVD+knn	0.662	0.698	0.671	0.850	0.586	0.650	0.686
	DWT+knn	0.642	0.639	0.671	0.837	0.590	0.590	0.661
	GMLS+knn	0.593	0.706	0.684	0.821	0.622	0.708	0.689
	LMLS+knn	0.713	0.802	0.708	0.828	0.617	0.641	0.718
K=20	EUC+knn	0.562	0.750	0.693	0.779	0.511	0.547	0.640
	ERP+knn	0.639	0.704	0.678	0.828	0.643	0.644	0.689
	EDR+knn	0.615	0.753	0.735	0.862	0.569	0.679	0.702
	SVD+knn	0.670	0.702	0.698	0.844	0.595	0.634	0.690
	DWT+knn	0.638	0.661	0.669	0.847	0.587	0.609	0.669
	GMLS+knn	0.617	0.698	0.692	0.819	0.642	0.717	0.698
	LMLS+knn	0.708	0.787	0.722	0.828	0.597	0.653	0.716

for all the methods. The corresponding optimal performance for each method is not significantly better than that is obtained based on only the feature *group*. It indicates *group* is a very important feature in collaborative online games.

Our proposed methods GMLS and LMLS nearly achieve all the optimal performance for these six features, and LMLS achieves the best average F1 score. It indicates the effectiveness of our proposed algorithms. Different from EDR and ERP, GMLS and LMLS do not explicitly consider local time shifting in the original time series. GMLS and LMLS first project time series into a new space where the task-specific objective function can achieve (local) optimal (i.e., Eq. 3 and 4), and they have more flexibility to adapt to specific tasks. Our methods provide a principled approach to incorporate task-specific information into time series similarity measures while general measures cannot do it at all.

Since the performance of k -NN classifier depends on the size of training set, we vary the size of training set to see how it affects different methods. We fix $k = 10$ in the

Table 3. Performance comparison of average F1 score with varying training set sizes

size	EUC+knn	ERP+knn	EDR+knn	GMLS+knn	LMLS+knn
1 fold	0.598	0.648	0.677	0.635	0.665
2 folds	0.627	0.675	0.691	0.662	0.709
3 folds	0.641	0.696	0.710	0.664	0.717
4 folds	0.639	0.677	0.697	0.682	0.720

k -NN classifier since we have found that it gives corresponding optimal performance for all these methods. Recall we have divided the entire data set into five folds, we random select one fold as test data and then vary the size of training set. We present the results in Table 3. We can observe that LMLS still performs best in terms of average F1 score and the performance increases with more training data. An interesting thing is that baselines achieve their best with 3-fold training data. A possible explanation is that these baselines are general measures which do not rely on training data so that we may not obtain better performance with more training data.

Efficiency Analysis. In practice, efficiency is an important issue for time series measures. DTW, ERP, LCSS and EDR are dynamic programming ways to compute the distance between two time series. Given two time series of length L , these algorithms have the same time complexity: $O(L^2)$; our methods GMLS and LMLS have the time complexity $O(L^2l + l)$, where l is the dimension number in the projected space and usually much smaller than the original length L . For GMLS and LMLS, we have the additional cost to learn transformation matrix \mathbf{W} . Given the size of training set is N , the time complexity for learning \mathbf{W} is $O(N^2L + L^3)$ for GMLS and $O(N^2L + N \log N + L^3)$ for LMLS, where $O(L^3)$ is the time cost for solving eigenvectors. Although the cost is high when L is large, given a task, we can learn \mathbf{W} in an offline way and compute the time series similarity in an efficient online way. In our task, for each feature, we need to learn a particular transformation matrix \mathbf{W} for $992 \times \frac{4}{5} = 793$ time series in training set. We implement LMLS and GLMS in Python, and the total running time to learn \mathbf{W} on the entire data set for six feature are 252.2 seconds and 577.5 seconds respectively for GLMS and LMLS.

Qualitative Analysis of Our Method. To get an intuitive idea of why our method works, we select four example time series: two are from normal users while the other two are from churn users. Given a time series \mathbf{x} , we can project it to a low-dimension space, i.e., $\mathbf{x}' = \mathbf{W}^T \mathbf{x}$. We set the projected dimension number to 10. We present the original times series and the corresponding transformed time series in Figure 1. We can see that our methods clearly capture a few important discriminative patterns for both classes. We also present the pairwise distance before and after the transformation in Table 4. We can see that after transformation, our method results in much smaller intra-class distance while inter-class distance does not change too much.

Parameter Tuning. There are two parameters to tune in LMLS: size of local neighborhood k_{local} ⁵ and projected space dimension number l . We first find the optimal parameter setting using a grid search method: $k_{local} = 10$ and $l = 5$. Then we fix one

⁵ Recall LMLS first builds a local neighborhood for each example, and it is not the k in outer k -NN classifier.

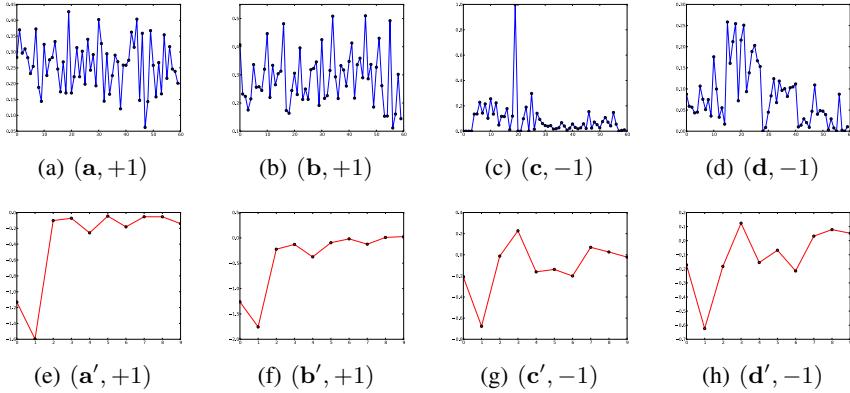


Fig. 1. Example time Series before and after the transformation via metric learning. **a**, **b**, **c** and **d** are original time series, **a'**, **b'**, **c'** and **d'** are transformed time series using metric learning respectively. We attach the corresponding class labels of these four time series.

Table 4. Comparison of inter-class and intra-class time series distance using Euclidean distance

(a) Original time series.

	(a, +1)	(c, -1)
(b, +1)	0.987	1.628
(d, -1)	1.994	1.171

(b) Transformed time series.

	(a', +1)	(c', -1)
(b', +1)	0.374	1.409
(d', -1)	1.600	0.242

parameter and then vary the other. We present the results in Figure 2. In Figure 2(a), we can see our method generally works well when $k_{local} > 5$. In Figure 2(b), we can see our method achieve good performance when l is set to a small value, e.g., 5 and 10.

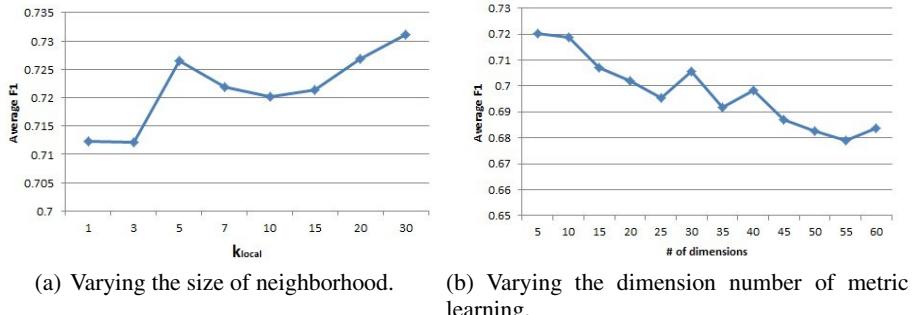


Fig. 2. Parameter tuning for LMLS

Dealing with Unequal-Length Time Series. Until now, we only focus on time series of the same length. In practice, the length of time series can be varying. In this part,

Table 5. Comparison of F1 scores for different methods on unequal-length time series. We adopt the corresponding optimal parameter setting in Table 2 for these methods.

Methods	completed	logTime	logCount	group	general	money	Average
ERP+knn	0.609	0.716	0.683	0.722	0.589	0.641	0.660
EDR+knn	0.579	0.713	0.719	0.848	0.532	0.604	0.666
GMLS+knn	0.565	0.684	0.622	0.814	0.634	0.727	0.674
LMLS+knn	0.747	0.767	0.732	0.827	0.591	0.566	0.705

we examine how the performance of our methods on unequal-length time series. We randomly select $\frac{1}{3}$ of examples in training set and test set. We remove the first 20 entries in the original time series, so these time series only have 40 entries. We select ERR and EDR as comparisons since they perform well and can deal with unequal-length time series. Since metric learning cannot deal with unequal input in essence, we turn to a heuristic method. We add 20 zero entries at the beginning of these 40-length time series, therefore these time series have the length of 60 again. We present the results in Table 5. We can see that all the methods perform worse compared to the performance on equal-length time series, but our proposed method LMLS still perform best in terms of average F1 score. It indicates the robustness of LMLS. Indeed, LMLS does not capture simple surface patterns but can reveal the discriminative patterns in the latent space. We will investigate more principled methods to deal with unequal-length time series in the future.

5 Conclusion

In this paper, we investigate how to develop task-specific time series similarity measures. We adopt metric learning as the principled approach to evaluate time series similarity. The major merit of metric learning is that we can seek a distance function which helps improve the performance of the specific task. We propose two metric learning based methods: one is based on global metric learning and the other is based on local metric learning. We use the data from a large online game and study the problem of user churn prediction. Given a user, we extract the information of six prominent features from user logs and build corresponding time series for each feature in a two-month time span. Our task is defined to be time series classification, i.e., given a time series from a user, we want to predict whether it is from a normal user or a user who would churn later. We adopt k -NN as the classifier and select a few widely used similarity measures as comparisons. Extensive experiments show that our proposed methods are very effective.

Although there are a large body of studies on evaluating time series similarity, they seldom consider task-specific information, e.g., supervised information, when developing similarity measures. Our study provides a promising solution to this problem and we believe our work will inspire more follow-up studies.

There can be two promising directions to improve current work. First, in real online games, the data set is usually imbalanced, we can consider designing the new objective function which can deal with imbalanced classification, e.g., incorporating class weights. Secondly, the current work can be extended to deal with semi-supervised classification in the case when very few labeled instances are available.

Acknowledgments. The authors are partially supported by RenRen Games Grant QXWJ-YX-201206017 and NSFC Grant 61272340, 61073082. Xin Zhao was supported by Microsoft PhD Fellowship (Asia). We thank the three anonymous reviewers for their insightful comments.

References

1. Berndt, D.J., Clifford, J.: Using Dynamic Time Warping to Find Patterns in Time Series. In: KDD Workshop (1994)
2. Vlachos, M., Gunopoulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: Proceedings of the 18th International Conference on Data Engineering, ICDE 2002 (2002)
3. Chen, L., Özsu, M.T., Oria, V.: Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, SIGMOD 2005 (2005)
4. Xing, Z., Pei, J., Keogh, E.: A brief survey on sequence classification. SIGKDD Explor. Newsl. 12(1) (November 2010)
5. Wang, F., Chen, S., Zhang, C., Li, T.: Semi-supervised metric learning by maximizing constraint margin. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008 (2008)
6. Chen, L., Ng, R.: On the marriage of l_p -norms and edit distance. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, vol. 30 (2004)
7. Korn, F., Jagadish, H.V., Faloutsos, C.: Efficiently supporting ad hoc queries in large datasets of time sequences. SIGMOD Rec. 26(2) (June 1997)
8. Pong Chan, K., Fu, A.W.C.: Efficient time series matching by wavelets. In: Proceedings of 15th International Conference on Data Engineering, ICDE 1999 (1999)
9. Wei, L., Keogh, E.: Semi-supervised time series classification. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006 (2006)
10. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: A survey and empirical demonstration. Data Min. Knowl. Discov. 7(4) (October 2003)
11. Nguyen, M.N., Li, X.L., Ng, S.K.: Positive unlabeled learning for time series classification. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI 2011, vol. 2 (2011)
12. Wang, F., Sun, J., Ebadollahi, S.: Composite distance metric integration by leveraging multiple experts' inputs and its application in patient similarity assessment. Statistical Analysis and Data Mining 5(1) (2012)
13. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. 10 (June 2009)
14. Wang, F.: Semisupervised metric learning by maximizing constraint margin. IEEE Transactions on Systems, Man, and Cybernetics, Part B 41(4), 931–939 (2011)
15. Korn, F., Jagadish, H.V., Faloutsos, C.: Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences. In: ACM SIGMOD, pp. 289–300 (May 1997)

Using Coalitional Games to Detect Communities in Social Networks

Lihua Zhou¹, Chao Cheng¹, Kevin Lü², and Hongmei Chen^{1,*}

¹ Department of Computer Science and Engineering, Yunnan University,
Kunming 650091, China

² Brunel University, Uxbridge, UB8 3PH, UK
{lhzhou, hmchen}@ynu.edu.cn, 136388340@qq.com,
jiang.jianglu@gmail.com

Abstract. The community detection in social networks is important to understand the structural and functional properties of networks. In this paper we propose a coalitional game model for community detection in social networks, and use the Shapley Value in coalitional games to evaluate each individual's contribution to the closeness of connection. We then develop an iterative formula for computing the Shapley Value to improve the computation efficiency. We further propose a hierarchical clustering algorithm GAMEHC to detect communities in social networks. The effectiveness of our methods is verified by preliminary experimental result.

Keywords: social networks, community detection, game theory.

1 Introduction

With the increasing popularity of social networks, community detection has received a great deal of attentions recently. Researchers have proposed various methods to detect communities based on principles of physics, statistics, artificial intelligence, etc. These methods reveal structures of networks from different perspectives. However, due to the complex of social networks, some new requirements raise for community detection algorithms. Meanwhile, many networks are known to possess hierarchical organization, where communities are recursively grouped into a hierarchical structure. Existing algorithms offer no solution to the combination of these requirements. Therefore, developing new community detection approaches for large networks is a challenging task.

Coalitional game (Zlotkin and Rosenschein 1994) focuses on the cooperative behaviors of groups of players, while each group of players is called a coalition. By cooperation, players gain more payoff than those individuals on their own. In coalitional games, the Shapley Value is a classical normative solution concept because it provides a unique and fair solution (Shapley 1953).

* Corresponding author.

In social networks, co-operations amongst individuals co-exist with conflicts amongst individuals, because individuals with similar interests are more likely to cooperate with each other, but an individual's influence in a network is dependent on itself as well as others. These interactive and cooperative features can be analyzed by using game theory. Furthermore, a community in a social network is a group of individuals with closely connection. Clearly, the degree of 'closely connection' depends on the interrelationships between individuals. Therefore, communities can be described by coalitions, and each individual's contribution to the closeness of connection can be evaluated by the Shapley Value.

In this paper, we propose coalitional game models for community detection based on topological structures of social networks, and we use the Shapley Value to evaluate each individual's contribution to the closeness of connection. It is known that the computation of the Shapley Values is not polynomial time (Lemke and Howson 1964), thus we develop an iterative formula to improve the computation efficiency. We also present a hierarchical clustering algorithm GAMEHC for detecting communities and apply it into a real network.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 introduces the **coalitional** game theoretic approach for community detection. Section 4 shows experimental results. Section 5 concludes the paper.

2 Related Work

Fortunato (2010) provided a comprehensive survey on community detection approaches, such as hierarchical clustering, modularity optimization, percolation, etc. In addition, many new methods have been proposed to detect communities in recent years (Ahn et al. 2010; Lin et al. 2011; Shi et al. 2012; Zhao et al. 2012).

Coalitional game theory has been used in various applications, for example, Liu et al. (2011) applied coalitional game theory to discover different groups from given objects; Saad et al. (2009) applied coalitional game theory to the analysis of communications in wireless networks; Moretti et al. (2010) use coalitional game theory on biological networks to measure centrality and power of genes.

The Shapley Value is a key solution concept for coalitional games, due to it provides a unique and fair solution. However, for many coalitional games, the computation of the Shapley Values is very heavy., so many approximation methods have been developed (Owen 1972; Fatima et al. 2008). In the context of networks, Karthik et al. (2010) developed exact analytical formulas for computing the Shapley Value based centralities.

3 A Coalitional Game Approach for Community Detection

3.1 Coalition Games and the Shapley Value

Definition 1. *Coalitional games* (Liu et al. 2007). A *coalitional game* is a pair (N, v) , where $N = \{1, 2, \dots, n\}$ denotes a finite set of players and $v: 2^N \rightarrow R$ is

the *characteristic function*, assigning a real value to each $S \subseteq N$. S (i.e. a group of players) is called a *coalition* and $v(S)$ is called the *worth* of this coalition.

In coalitional games, how to split the total worth between the players is an important issue. The *Shapley Value* (Shapley 1953) is a widely used value division scheme in the theory of coalitional games.

Definition 2. The Shapley Value (Shapley 1953). The Shapley Value of player i in coalition $S \subseteq N$ with respect to (N, v) is given by formula (1):

$$\forall i, SH_v(S, i) = \sum_{\{T \subseteq S | i \in T\}} \frac{(|T|-1)!(|S|-|T|)!}{|S|!} [v(T) - v(T - \{i\})] \quad (1)$$

Where $|S|$ (resp. $|T|$) is the cardinality of the set S (resp. T), $v(T) - v(T - \{i\})$ defines the marginal contribution of player i to the coalition T . $\frac{(|T|-1)!(|S|-|T|)!}{|S|!}$ denotes the probability distribution function of subsets of S .

3.2 A Coalitional Game Model for Community Detection

Definition 3. A coalitional game model for community detection. Let $N = \{1, 2, \dots, n\}$ be the set of all individuals in a social network, $A = (a_{ij})_{n \times n}, i, j \in N$ be the adjacency matrix, where $a_{ij} = 1, i, j \in N$ if a relationship exists between vertex i and j , otherwise $a_{ij} = 0, i, j \in N$, then the coalitional game model for community detection is defined as $CCG = (N, v)$, where individual $i \in N$ is a player participating the coalitional game, and the characteristic function $v(S)$ is defined by following formula (2):

$$v(S) = \begin{cases} v(S) = 0, & S \subseteq N, |S| = 1, \text{ or } S = \Phi \\ v(S) = \sum_{i \in S} \sum_{\substack{j \in S \\ j \neq i}} \frac{a_{ij}}{d(i)}, & S \subseteq N, |S| \geq 2, d(i) \neq 0 \end{cases} \quad (2)$$

Where $d(i) = \sum_{j \in N} a_{ij}$, it is the degree of individual i . $v(S)$ satisfies Theorem 1~3.

Theorem 1. $v(S)$ satisfies Superadditive, i.e. $\forall S_1, S_2 \subseteq N$, if $S_1 \cap S_2 = \Phi$, then $v(S_1) + v(S_2) \leq v(S_1 + S_2)$.

Theorem 2. $v(S)$ satisfies: if $S_1, S_2 \subseteq N, S_1 \subseteq S_2$, then $v(S_2) \geq v(S_1)$.

Theorem 3. If $S \subseteq N$, $i \in S$, then $v(S) - v(S - \{i\}) = \sum_{j \in S - \{i\}} \left(\frac{a_{ij}}{d(i)} + \frac{a_{ji}}{d(j)} \right)$.

The proofs for all Theorems are omitted by the limitation of space.

3.3 Efficient Computation of the Shapley Value for Community Detection

In coalition games, the computation of the Shapley Value is crucial. However, the complexity of computing Shapley Value is heavy. Based on the Definition 3, we can develop an iterative formula for computing the Shapley Value to improve the computation efficiency. The iterative formulas for computing the Shapley Value are given in the Theorem 4.

Theorem 4. Given a coalitional game model for community detection (N, v) , $\forall S_1, S_2 \subseteq N$, $S = S_1 + S_2$, then the Shapley Value of player $i \in S_1$ and $j \in S_2$ in coalition S with respect to (N, v) can be compute by formula (3):

$$\begin{aligned} SH_v(S, i) &= SH_v(S_1, i) + \frac{1}{2} \sum_{j \in S_2} \left(\frac{a_{ij}}{d(i)} + \frac{a_{ji}}{d(j)} \right), \quad i \in S_1 \\ SH_v(S, j) &= SH_v(S_2, j) + \frac{1}{2} \sum_{i \in S_1} \left(\frac{a_{ij}}{d(i)} + \frac{a_{ji}}{d(j)} \right), \quad j \in S_2 \end{aligned} \quad (3)$$

3.4 The Algorithm for Detecting Communities

Theorem 4 means that if we combine any two small-size coalitions to a large-size coalition, then the Shapley Value w.r.t. the large-size coalition will not be lesser than those of small-size coalitions. Based on this property, we propose a hierarchical clustering algorithm GAMEHC to generate large-size coalition level after level. With the process of combination, multiple partitionings will be produced. These partitionings provides a rich hierarchy of structure, and reveal hierarchy organization of players. The GAMEHC algorithm is described as follows:

Algorithm GAMEHC

Input :

$N = \{1, 2, \dots, n\}$, the set of players (individuals),

$A = (a_{ij})_{n \times n}, i, j \in N$, the adjacency matrix

Output : communities

Step 1. Initialization

$l = 1; S^l = \{\Phi\}; // l : a level number; S^l : the set of coalitions$

For $i = 1$ to n

$S_i^l = \{i\}; S^l = S^l \cup \{S_i^l\}; // S_i^l : the ith coalition in S^l$

End for

Step 2. Form large-size coalitions

While $|S^l| > 1$

$l = l + 1;$

For each $S_i^{l-1} \in S^{l-1}$

```

if  $\forall x \in S_i^{l-1}$ ,  $SH_v(S_i^{l-1} \cup S_j^{l-1}, x) \geq SH_v(S_i^{l-1} \cup S_k^{l-1}, x)$ , where
 $S_j^{l-1}, S_k^{l-1} \in S^{l-1}, j \neq k$ , or  $S_k^{l-1} = \Phi$ ,
then  $S_r = S_i^{l-1} \cup S_j^{l-1}$ ,  $S^l = S^{l-1} - \{S_i^{l-1}\} - \{S_j^{l-1}\}$ ,  $S^l = S^l + \{S_r\}$ 
end if
end For
End while

```

Step 3. Output elements of $S' \cup l'$ is a level set by user.

4 Experiments and Results

We applied the GAMEHC algorithm to detect communities in Zachary's karate network (Zachary 1977), shown in Fig.1. It is a well-known social network used as a benchmark for evaluating community detection algorithms.

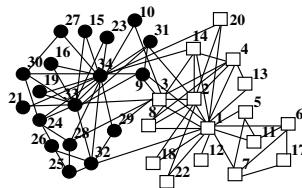


Fig. 1. Zachary's karate network

The process of combining coalitions is represented by a dendrogram, shown in Fig.3.



Fig. 2. The dendrogram of Zachary's karate network

From Fig.2, we can see that the process of combining coalitions reveals the hierarchy organization of the network. Each individual selects a coalition to join in some level, such that it can receive the highest Shapley Value. We can extract communities at multiple levels by cutting this dendrogram according to various modularity values.

5 Conclusion

Community detection is an important issue in social network analysis. In this paper a coalitional game theoretic approach for detecting communities has been presented. To

improve the computation efficiency, an iterative formula for computing the Shapley Value is developed, and last a hierarchical clustering algorithm GAMEHC is proposed. This algorithm can reveal hierarchy organization of networks.

Acknowledgments. This research was supported by the National Natural Science Foundation of China under Grant No.61262069, No.61063008, No.61163003, No.61272126, the Science Foundation of Yunnan Province under Grant No. 2010CD025, and the Yunnan Educational Department Foundation under Grant No.2012C103. Kevin Lü's work is partly supported by BRIGHT award.

References

1. Ahn, Y.Y., et al.: Link communities reveal multi-scale complexity in networks. *Nature* 466, 761–764 (2010)
2. Fatima, S.S., et al.: A linear approximation method for the Shapley Value. *Artificial Intelligence* 172, 1673–1699 (2008)
3. Fortunato, S.: Community detection in graphs. *Physics Reports* 486, 75–174 (2010)
4. Aadithya, K.V., Ravindran, B., Michalak, T.P., Jennings, N.R.: Efficient Computation of the Shapley Value for Centrality in Networks. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 1–13. Springer, Heidelberg (2010)
5. Lemke, C., Howson, J.: Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics*, 413–424 (1964)
6. Lin, Y.R., et al.: Community Discovery via Metagraph Factorization. *ACM Transactions on Knowledge Discovery from Data* 5(3), 17 (2011)
7. Liu, W.Y., et al.: An approach for multi-objective categorization based on the game theory and Markov process. *Applied Soft Computing* 11, 4087–4096 (2011)
8. Liu, W.Y., et al.: Intelligent Data Analysis. Science Press, Beijing (2007)
9. Moretti, S., et al.: Using coalitional games on biological networks to measure centrality and power of genes. *Bioinformatics* 26(21), 2721–2730 (2010)
10. Owen, G.: Multilinear extensions of games. *Management Science* 18(5), 64–79 (1972)
11. Saad, W., et al.: Coalitional game theory for communication networks: a tutorial. *IEEE Signal Processing Magazine* 26(5), 77–97 (2009)
12. Shapley, L.S.: A Value for N-person games. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to be Theory of Games, pp. 307–317. Princeton University Press (1953)
13. Shi, C., et al.: Multi-objective community detection in complex networks. *Applied Soft Computing* 12, 850–859 (2012)
14. Zachary, W.W.: An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* 33(4), 452–473 (1977)
15. Zhao, Z.Y., et al.: Topic oriented community detection through social objects and link analysis in social networks. *Knowledge-Based Systems* 26, 164–173 (2012)
16. Zlotkin, G., Rosenschein, J.: Coalition cryptography and stability mechanisms for coalition formation in task oriented domains. In: Association for the Advancement of Artificial Intelligence, pp. 432–437 (1994)

A Comparison Study of Clustering Models for Online Review Sentiment Analysis

Baojun Ma^{1,3}, Hua Yuan^{2,*}, and Qiang Wei¹

¹ School of Economics and Management, Tsinghua University, 100084 Beijing, China
{mabj.03,weiq}@sem.tsinghua.edu.cn

² School of Management and Economics, University of Electronic Science and Technology of China, 610054 Chengdu, China
yuanhua@uestc.edu.cn

³ School of Economics and Management, Beijing University of Posts and Telecommunications, 100876 Beijing, China

Abstract. In this work, we conduct a comparison study of the online review sentiment clustering problem from a combined perspective of data preprocessing, VSM modeling and clustering algorithm. To that end, we first introduce some methods for data preprocessing. Then, we explore the impacts of the term weighting models for review representation. Finally, we present detailed experiment results of some review clustering techniques. The conclusions would be valuable for both the study and usage of clustering methods in online review sentiment analysis.

Keywords: Online review, sentiment analysis, term weighting model, clustering algorithm.

1 Introduction

In recent year, online reviews have become an important resource for people to provide product information and recommendations from the customer perspective [1]. The customer reviews are so useful that almost all the e-commerce related organizations, such as Amazon and Google, have accumulated a huge amount of reviews data, and the analysis of these data to extract latent public opinion and sentiment is a challenging task. To automate the sentiment analysis, different approaches [2–4] in literature have been applied to predict the sentiments of words, expressions or documents.

Clustering, which tries to find the natural clusters in the data by calculating the distance from the centers of the clusters, is especially useful for organizing documents to improve information retrieval [5]. However, the analysis results generated by clustering method would be affect heavily by some intermediate steps, such as preprocessing strategy, term weighting model and clustering algorithm. It is valuable for online review sentiment analysis to make clear that: which kind of clustering algorithm is more effective for sentiment analysis? and

* Corresponding author.

which types of term weighting models are more suitable for review representation? In this work, we conduct comparison study of the online review sentiment clustering problem from a combined perspective of data preprocessing, VSM (Vector Space Model) modeling [6] and clustering algorithm.

2 Related Work

The task of sentiment analysis is to judge whether a review expresses a positive, neutral or negative opinion and a lot of efforts have been devoted into this area in literature [7]. The typical work is method that presented by Pang and Lee of sentiment classification on the document level [3].

Also, few efforts have been devoted to the study of sentiment analysis with clustering. Agarwal et al. Li & Liu [8] proposed a method to choose solid polarity reviews to generate a positive seed set and a negative seed set to solve this problem. Zhai et al. [9] studied the problem of product feature clustering for opinion mining applications, in which they casted the problem as a semi-supervised learning task. An approach of semi-automatic public sentiment analysis for opinion and district is proposed in [10], which includes automatic data acquiring, sentiment modeling, opinion clustering, and district clustering, and manual threshold setting and result analysis. [11] investigated the effect of feature weighting on document clustering, including a novel investigation of Okapi BM25 feature weighting. Especially, [12] presented the results of some common document clustering techniques. However, there are not impressive research on comparing the different clustering performance under various environments.

3 The Methodology

The research framework consists of four parts: data preprocessing, VSM modeling, clustering and results evaluation.

3.1 Data Preprocessing

First, a part-of-speech tagger developed by Stanford University is used to tag the reviews [13]. Under the impact of the work of the first step, the words which are not tagged as being either an adjective or adverb would be eliminated. Then, the words stemming is done by applying Porter's algorithm [14]. Finally, we utilize the stop-word list to remove stop words which were built by Gerard Salton and Chris Buckley for the experimental SMART information retrieval system.

3.2 Term Weighting Models

Let $D = \{d_1, d_2, \dots, d_N\}$ be a set of documents/reviews and $T = \{t_1, t_2, \dots, t_M\}$ be the complete term set of D , in which, $d_i = [w_{i1}, w_{i2}, \dots, w_{im}]$ where w_{ij} is the weight of term t_j to document d_i . Table 1 illustrates six term weighting models selected to be utilized in this study.

Table 1. Term weighting models used in our experiments

Weighting	Equation	Reference
Binary	$w_{ij} = \begin{cases} 1, & \text{if } tf_{ij} > 0; \\ 0, & \text{otherwise.} \end{cases}$	B. Ricardo et. al,(1999)
TF	$w_{ij} = tf_{ij}$	Salton et.al.(1981; 1983)
TF-IDF	$w_{ij} = tf_{ij} \times \log \frac{N}{df_j}$	Jones(1972)
BM25	$w_{ij} = \frac{tf_{ij}(k_1+1)}{tf_{ij} + k_1((1-b)+b \cdot \frac{dl_i}{avgdl})} \times \log \frac{N}{df_j}$	S.E. Robertson et. al,(1994)
DPH_DFR	$w_{ij} = \begin{cases} 0, & \text{if } tf_{ij} = 0; \\ \frac{(1 - \frac{tf_{ij}}{dl_i})^2}{tf_{ij} + 1} \cdot \{tf_{ij} \cdot \log(\frac{tf_{ij} \cdot avgdl}{dl_i} \cdot \frac{N}{tf_j}) \\ + 0.5 \log[2\pi \cdot tf_{ij} \cdot (1 - \frac{tf_{ij}}{dl_i})]\}, & \text{otherwise.} \end{cases}$	G. Amati, et. al,(2007)
H_LM	$w_{ij} = \log(1 + \frac{\lambda \cdot tf_{ij} \cdot sigmatf}{(1-\lambda) \cdot df_j \cdot dl_i})$	Hiemstra (2001)

3.3 Clustering Algorithms and Sentiment Recognition

We have selected a collection of 18 clustering algorithms to conduct a relative complete comparison. Table 2 lists all these algorithms in detail.

Table 2. The clustering algorithms used in our experiments

Algorithm	Short	Reference
K-means	Kmeans (Direct-I2)	Lloyd (1982)
Repeated bisecting k-means	RB-Kmeans (RBR-I2)	Steinbach, et al. (2000)
Partition around medoids	PAM	Kaufman & Rousseeuw (1990)
Clustering Large Applications based upon RANdomized Search	CLARANS	R. T. Ng & Han (2002)
Unnormalized spectral	Spect-Un	Luxburg (2007)
Random walk spectral	Spect-RW	Shi & Malik (2000)
Symmetric spectral	Spect-Sy	A. Y. Ng et al. (2001)
Principle component analysis + K-means	PCA-Kmeans	Pearson (1901)
Non-negative matrix factorization	NC-NMF	Xu et al. (2003)
Unweighted pair group method	UPGMA (Agglo-upgma)	Kaufman & Rousseeuw (1990)
Single linkage	Slink (Agglo-Slink)	Jain et al. (1999)
Complete linkage	Clink (Agglo-Clink)	Jain et al. (1999)
Repeated bisecting H1 with global optimization	RBR-H1	Zhao and Karypis (2004)
Direct H1	Direct-H1	Zhao and Karypis (2004)
Agglomerative I2	Agglo-I2	Zhao et al. (2005)
Agglomerative H1	Agglo-H1	Zhao et al. (2005)
Agglomerative single-link	cluster-weighted	Zhao et al. (2005)
Agglomerative complete-link	cluster-weighted	Zhao et al. (2005)

3.4 Evaluation

To evaluate the clustering effectiveness, a confusion matrix could be constructed as shown in Table 3. Cluster 1 is the positive cluster if $(a + d) \geq (b + c)$. Otherwise, Cluster 2 is the positive cluster. Consequently, $Accuracy = \frac{a+d}{a+b+c+d}$, if $(a + d) \geq (b + c)$; else $Accuracy = \frac{b+c}{a+b+c+d}$.

Table 3. The confusion matrix

	Cluster 1 Cluster 2	
Actual # of positive reviews	a	b
Actual # of negative reviews	c	d

4 Experimental Results

In this work, we introduce eight datasets in the following experiments (Table 4).

Table 4. The benchmark datasets

ID	Data set	URL
D1	Polarity Dataset V2.0	www.cs.cornell.edu/People/pabo/movie-review-data
D2	Sentence Polarity Dataset v1.0	www.cs.cornell.edu/People/pabo/movie-review-data
D3	Amazon reviews (Books)	
D4	Amazon reviews (DVDs)	www.cs.jhu.edu/~mdredze/datasets/sentiment/
D5	Amazon reviews (Electronics)	
D6	Amazon reviews (Kitchen)	
D7	TripAdvisor-15763	http://patty.isti.cnr.it/~baccianella/reviewdata/corpus/
D8	Amazon-83713	http://patty.isti.cnr.it/~baccianella/reviewdata/corpus/

4.1 Results on Term Weighting Models

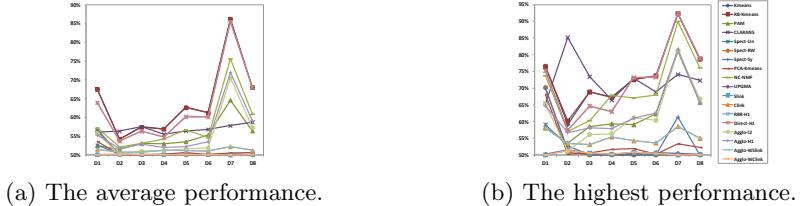
We compared the clustering results on six weighting models (the results are shown in Table 5). We see that, on average, BM25, DPH_DFR and H_LM weighting models are somewhat better than TF_IDF, and notably better than Binary as well as TF. However, TF_IDF is actually sometimes better than H_LM and DPH_DFR (D7), and performs similar results to BM25 on D2 and D7.

Table 5. The percentage difference in average accuracy

Dataset	Binary	TF	TF_IDF	BM25	DPH_DFR	H_LM	Max
D1	-3.12%	-5.16%	-8.17%	-0.95%	0	-5.91%	0.585
D2	-1.65%	-1.39%	-0.63%	-0.45%	0	-0.60%	0.521
D3	-0.42%	-2.09%	-0.25%	0	-0.19%	-0.79%	0.529
D4	-1.69%	-1.48%	-0.82%	0	-0.62%	-0.98%	0.528
D5	-1.83%	-5.15%	-3.01%	-2.92%	0	-1.80%	0.552
D6	-3.69%	-4.57%	-2.15%	-0.51%	-0.18%	0.00%	0.548
D7	-5.10%	-6.05%	-0.05%	0	-1.96%	-1.71%	0.649
D8	-2.33%	-4.08%	-2.72%	0	-0.79%	-1.12%	0.576
Overall	-2.07%	-3.36%	-1.76%	-0.10%	0.00%	-1.16%	0.558

4.2 Results on Clustering Algorithms

In this section, we try to find out which kinds of clustering algorithms are more effective for clustering-based sentiment analysis. The results are illustrated in Fig. 1, in which the lines of Kmeans and RB-Kmeans are almost coinciding with each other, so as are those of RBR-H1 and Direct-H1.

**Fig. 1.** The average (a) and highest (b) performance of each clustering algorithm

To see more clearly the differences between clustering performances, we make a comparison of the 18 clustering algorithms by dataset and average over all clustering algorithms for that dataset. Fig. 1 and Table 6 indicate that, on average, four algorithms of Kmeans, RB-Kmeans, RBR-H1 and Direct-H1 show clear advantage over the other 14 methods on clustering accuracy. However, CLARANS algorithm is actually somewhat better than the above four best methods and substantially better than other 13 algorithms for D2 and D3.

Table 6. The percentage difference in average accuracy

Algorithm	D1	D2	D3	D4	D5	D6	D7	D8	Overall
Kmeans	0	-3.9%	0	0	0	0	0	0	0.0%
RB-Kmeans	0	-3.7%	-0.2%	0	-0.2%	0	0	0	0.0%
PAM	-22.4%	-8.5%	-7.7%	-7.0%	-14.7%	-9.6%	-25.1%	-17.1%	-14.5%
CLARANS	-17.0%	0	0	-2.5%	-10.0%	-7.2%	-32.9%	-13.7%	-11.5%
Spect-Un	-22.1%	-10.8%	-12.9%	-12.0%	-20.1%	-18.1%	-41.9%	-26.3%	-21.5%
Spect-RW	-16.7%	-11.0%	-12.9%	-12.0%	-20.1%	-18.1%	-41.9%	-26.3%	-20.9%
Spect-Sy	-15.9%	-11.0%	-12.9%	-12.0%	-20.1%	-18.1%	-41.6%	-26.3%	-20.7%
PCA-Kmeans	-20.6%	-11.0%	-12.9%	-11.8%	-19.3%	-18.1%	-41.4%	-25.6%	-21.1%
NC-NMF	-15.3%	-7.5%	-7.8%	-4.7%	-9.6%	-10.8%	-12.3%	-10.4%	-9.7%
UPGMA	-25.8%	-11.0%	-12.9%	-12.0%	-19.9%	-18.1%	-42.0%	-26.3%	-22.0%
Slink	-25.8%	-10.8%	-12.9%	-12.0%	-20.1%	-18.1%	-42.0%	-26.3%	-22.0%
Clink	-23.7%	-9.9%	-11.8%	-10.0%	-18.5%	-16.7%	-39.4%	-24.7%	-20.3%
RBR-H1	-5.2%	-4.6%	-2.1%	-3.9%	-3.8%	-1.6%	-0.8%	0	-2.2%
Direct-H1	-5.3%	-4.6%	-2.1%	-3.9%	-4.1%	-1.8%	-0.8%	0	-2.3%
Agglo-I2	-17.3%	-10.8%	-11.3%	-10.2%	-17.7%	-15.2%	-18.2%	-15.7%	-14.5%
Agglo-H1	-18.1%	-8.2%	-8.2%	-8.6%	-16.7%	-12.6%	-16.5%	-13.1%	-12.7%
Agglo-Wslink	-23.7%	-9.9%	-11.8%	-10.0%	-18.5%	-16.7%	-39.4%	-24.7%	-20.3%
Agglo-Wclink	-25.8%	-10.8%	-12.9%	-12.0%	-20.1%	-18.1%	-42.0%	-26.3%	-22.0%
Max	0.675	0.563	0.575	0.569	0.627	0.612	0.862	0.68	0.643

5 Conclusion

In this work, we find averagely the following experimental conclusions for online review sentiment clustering:

- BM25, DPH_DFR and H_LM weighting models are somewhat better than TF_IDF, and notably better than Binary as well as TF.
- Kmeans, RB-Kmeans, RBR-H1 and Direct-H1 show clear advantage over the other 14 methods on clustering accuracy. However, CLARANS algorithm is actually somewhat better than the above four best methods and substantially better than other 13 algorithms for D2 and D3.

The experiment methods and conclusions would be valuable for both the study and usage of clustering methods in online review sentiment analysis.

Acknowledgments. The work was partly supported by the National Natural Science Foundation of China (71271044/U1233118/71072015/71110107027), and the Tsinghua University Initiative Scientific Research Program (20101081741).

References

1. Zhu, F., (Michael) Zhang, X.: Impact of online consumer reviews on sales: The moderating role of product and consumer characteristics. *Journal of Marketing* 74(2), 133–148 (2010)
2. Yi, J., Nasukawa, T., Niblack, W., Bunescu, R.: Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In: *Proceedings of the ICDM 2003*, Florida, USA, pp. 427–434 (2003)
3. Pang, B., Lee, L.: Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1-2), 1–135 (2008)
4. Prabowo, R., Thelwall, M.: Sentiment analysis: A combined approach. *Journal of Informetrics* 3, 143–157 (2009)
5. Anick, P., Vaithyanathan, S.: Exploiting Clustering and Phrases for Context-Based Information Retrieval. In: *Proceedings of the 20th ACM SIGIR*, pp. 314–323 (1997)
6. Salton, G., Wong, A., et al.: A vector space model for automatic indexing. *Communication of the ACM* 18(11), 613–620 (1975)
7. Liu, B.: *Sentiment Analysis and Opinion Mining*. Morgan and Claypool Publishers (May 2012)
8. Li, G., Liu, F.: Application of a clustering method on sentiment analysis. *Journal of Information Science* 38(2), 127–139 (2012)
9. Zhai, Z., Liu, B., Xu, H., Jia, P.: Clustering product features for opinion mining. In: *Proceedings of the WSDM 2011*, New York, USA, pp. 347–354 (2011)
10. Wang, D., Feng, S., Yan, C., Yu, G.: An approach of semi-automatic public sentiment analysis for opinion and district. In: Wang, L., Jiang, J., Lu, J., Hong, L., Liu, B. (eds.) *WAIM 2011. LNCS*, vol. 7142, pp. 210–222. Springer, Heidelberg (2012)
11. Whissell, J.S., Clarke, C.L.: Improving document clustering using Okapi BM25 feature weighting. *Information Retrieval* 14(5), 466–487 (2011)
12. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: *KDD Workshop on Text Mining* (2000)
13. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: *Proceedings of the EMNLP/VLC 2000*, Hong Kong, pp. 63–70 (2000)
14. Porter, M.F.: An algorithm for suffix stripping. *Program* 14(3), 130–137 (1980)

TopicDSDR: Combining Topic Decomposition and Data Reconstruction for Summarization

Zhiming Zhang, Hongjie Li, and Lian'en Huang^{*}

Shenzhen Key Lab for Cloud Computing Technology and Applications
Peking University Shenzhen Graduate School, Shenzhen, Guangdong, P.R. China
`{zhangzm319,lihongjie99}@gmail.com, hle@net.pku.edu.cn`

Abstract. Multi-document summarization attempts to select the most important information to generate a compressed summary from a collection of documents. From the perspective of data reconstruction, a good summary may also well reconstruct the original documents. A document generally contains a variety of information centered around a main topic and covers different aspects of the main topic. In this paper we propose a novel model that combines data reconstruction and topic decomposition to summarize the documents, named TopicDSDR, which can not only best reconstruct the original documents but also capture the semantic similarity and main topics. We discuss two kinds of reconstructions: linear reconstruction and nonnegative reconstruction. We use the generalized Kullback-Leibler(KL) divergence as the loss function to evaluate the quality of summary for linear and nonnegative reconstruction and develop two new algorithms respectively. We conduct experiments on DUC2006 and DUC2007 summarization data sets, the experimental results demonstrate the effectiveness of our proposed methods.

Keywords: Multi-Document Summarization, Data Reconstruction, LDA, K-L Divergence.

1 Introduction

Multi-document summarization aims at creating a compressed version of a given document set on the same or similar topic. A well organized summary can greatly help users to reduce information overload. In general, document summarization can be classified into two categories: abstractive summarization and extractive summarization. Abstractive summarization produces summaries by arranging new sentences from the original documents, while the more widely used extractive summarization generates summaries directly by selecting salient sentences from the original documents. In this paper, we focus on the extractive multi-document summarization.

Data reconstruction, from the matrix view, can be seen as finding a lower dimensional matrix to well approximate the high dimensional matrix [5]. In a

* Corresponding author.

recent paper, He et al.[11] proposed a new summarization framework called Document Summarization based on Data Reconstruction(DSDR). They argued that a good summary should consist of those sentences that can best reconstruct the original document. The document set in the DSDR is represented in the form of sentence-by-term matrix. It aims to find a linear combination of a small sentence vector set (summary) to reconstruct the original sentence-by-term matrix. However, the number of sentences in the summary produced by the DSDR is relatively large which approximates to the original sentence number. Moreover, the quality of the summary produced by the DSDR is relatively poor. The main reason is that the DSDR algorithm bases on the sentence-by-term matrix which is very sparse and can't capture the semantic similarity. For example, two sentences with small difference are regarded as two totally different sentences.

A document generally contains a variety of information centered around a main topic and covers different aspects of the main topics [12]. Recently, probabilistic topic models [3][13] have been successfully introduced into document summarization [1][2]. In this paper we propose a novel model that combines data reconstruction and topic decomposition to summarize the documents, named TopicDSDR, which can not only best reconstruct the original documents but also capture the semantic similarity. We point out that it is not appropriate to perform data reconstruction directly on word space, since it can't discriminate words from different topics and is time consuming due to high dimension of word space. Thus, it is difficult to find a small sentence set based on word space to best reconstruct the original text. Here we discuss two kinds of reconstructions: topic linear reconstruction, which reconstructs the document by linear combinations of the selected sentences; topic nonnegative reconstruction, which allows only additive, not subtractive, combinations among the selected sentences [11]. For each reconstruction, we use the KL-divergence as the loss function to evaluate the quality of summary and develop effective algorithms to solve them. For the topic linear reconstruction, we introduce a greedy algorithm to minimize the loss function by forward stepwise selection that choose a sub-optimal sentence set to reconstruct the original documents. For the topic nonnegative linear reconstruction, we design a multiplicative updating algorithm which guarantees converging monotonically at least locally optimal solution.

The rest of the paper is organized as follows. We first introduce the related work in section 2. In section 3, we give a brief introduction of the DSDR framework. In section 4, we discuss our TopicDSDR in detail. Experiment results and analysis are presented in section 5, and finally section 6 concludes the paper.

2 Related Work

Document summarization has been well studied in literatures. Earlier studies conducted by [7][19] were based on simple features like term frequency, keyword, word position and etc.

In recent years, graph-based ranking algorithms such as PageRank [4] and HITS [14] have been proposed for multi-document summarization in [8][21]. The

graph-based algorithms used nodes to represent the sentences and edges to represent the similarity among the sentences and then selected the sentences according to their PageRank or HITS scores.

Topic decomposition has been introduced into document summarization to help select the salient sentences. Arora et al. [1] used Latent Dirichlet Allocation(LDA) [3] to capture the topics being covered by the documents and using the topic distribution as the basis of choosing the sentences to form the summary. HierSum proposed by [10] utilized a hierarchical LDA-style model to represent content specificity as a hierarchy of topic vocabulary distributions.

Matrix factorization methods have also been employed to summarize documents. Gong et al. [9] applied the Latent Semantic Analysis which used the Singular Value Decomposition(SVD) to discover semantically important sentences for summarization. Arora et al. [2] combined the LDA with SVD to capture the latent topics and the orthogonal representation of topics in the documents. They first used the LDA to break down the documents into topics. For each topic, a sentence-by-term matrix was generated with the probabilities between the sentence and the terms, then applied the SVD decomposition to select the salient sentences according to their singular values. Lee et al. [16] used the Non-negative Matrix Factorization(NMF) [15] to select sentences for summary. The NMF can get a sparse weight vector of the semantic feature vectors. The relevant score of a sentence was calculated based on the weight matrix and then chose the top-k scores to generate the summary.

Finding the optimal extractive summary can be seen as a combinatorial optimization problem which is NP-hard to solve [18]. One of the well-known methods is Maximal Marginal Relevance(MMR) [6] that used a greedy algorithm to select the most relevant sentences while avoid the redundancy by removing sentences similar to the already selected sentences. Global inference algorithms for document summarization have been studied in recent years [20][22]. They treated the summary task as an Integer Linear Programming (ILP) problem. The optimal algorithms they described are different from ours, since we directly optimize the feature space without similarity computing.

3 The DSDR Framework

He et al. [11] proposed the document summarization framework based on data reconstruction. We first introduce some definitions used in DSDR. Given a sentence by term matrix $V = [v_1, v_2, \dots, v_N]^T \in \mathbb{R}^{N \times D}$, each row of V is a weighted term frequency vector of a sentence. N is the sentence number of the corpus and D is the vocabulary size of the corpus. $X = [x_1, x_2, \dots, x_M]^T \subset V$ is the summary sentences selected by DSDR and $M < N$.

The DSDR's generation of a summary can be described in two parts: (1) For any sentence v_i in the document, DSDR selects the related sentences X from the candidate set to reconstruct the given sentence v_i by learning a reconstruction function for the sentence v_i :

$$v_i \approx f_i(X; a_i) \quad (1)$$

(2) For the entire document, DSDR attempts to find an optimal set of representative sentences X to approximate the entire document V , by minimizing the reconstruction error with the following objective function:

$$\min_{a_i} \sum_{i=1}^N \|v_i - f_i(X; a_i)\|^2 \quad (2)$$

$\|\cdot\|$ is the L_2 -norm. They proposed two kinds of reconstruction algorithms: linear reconstruction and nonnegative linear reconstruction.

3.1 Linear Reconstruction

The reconstruction function $f_i(X; a_i)$ can be defined as a linear function:

$$f_i(X; a_i) = \sum_{j=1}^M a_{ij} x_j \quad (3)$$

Then we get the objective function of linear DSDR with regularization:

$$\begin{aligned} & \min_{a_i} \sum_{i=1}^N \|v_i - X^T a_i\|^2 + \lambda \|a_i\|^2 \\ & \text{s.t. } X \subset V, |X| = M \text{ and } A = [a_1, a_2, \dots, a_N]^T \in \mathbb{R}^{N \times M} \end{aligned} \quad (4)$$

A greedy algorithm has been proposed, see [11] for detail.

3.2 Nonnegative Linear Reconstruction

The parameter a_{ij} in linear reconstruction may be negative value, which means redundant information needs to be removed from the summary sentence set X . They tackled this problem by restricting a_{ij} to be nonnegative. The optimal objective function can be defined as :

$$\begin{aligned} & \min_{a_i, \beta_j} \sum_{i=1}^N \{\|v_i - V^T a_i\|^2 + \sum_{j=1}^N \frac{a_{ij}^2}{\beta_j}\} + \lambda \|\beta\|_1 \\ & \text{s.t. } \beta_j \geq 0, \quad a_{ij} \geq 0 \quad \text{and} \quad a_i \in \mathbb{R}^N \end{aligned} \quad (5)$$

where $\beta = [\beta_1, \dots, \beta_N]^T$ is an auxiliary variable to control the sentence selection. $\|\cdot\|_1$ is the L_1 norm. If $\beta_j = 0$, then all a_{1j}, \dots, a_{Nj} should be zero, which means the j -th candidate is not selected. A global optimal solution was proposed in [11].

4 Improving DSDR with Topic Decomposition

In this section, we first introduce the topic decomposition and the loss function based on KL divergence. Then we discuss two kinds of TopicDSDR algorithms: topic linear reconstruction and topic nonnegative reconstruction.

4.1 Topic Decomposition via Latent Dirichlet Allocation

LDA proposed by [3] is a generative three-level hierarchical bayesian probability model for discrete data such as documents. It models documents using a latent topic layer. In LDA, for each document d , first a multinomial distribution $\theta^{(d)}$ over topics is sampled from a Dirichlet prior distribution with hyper parameter α . Then, for each word w_{di} in document d , a topic z_{di} is sampled from the multinomial distribution $\theta^{(d)}$. Finally, the word is sampled from the distribution over words $\phi^{(z)}$ with a specific topic z . Therefore, the probability a word w is generated from the given document d is :

$$p(w|d, \alpha, \beta) = \sum_{i=1}^T p(w|z_i, \beta)p(z_i|d, \alpha) \quad (6)$$

In our work, we treat each sentence as a document. LDA is utilized to detect the topic distribution of sentences. We apply GibbsLDA++ [23] which using the Gibbs sampling to do the topic decomposition. After iteration, we can get two probability distributions: the topic distribution of sentences θ and the word distribution of topics ϕ . We use the topic distribution of sentences θ as the sentence-by-topic matrix V , each row of V is sum to one. Our TopicDSDR is conducted on this matrix.

4.2 Loss Function of TopicDSDR

In order to find a reconstruction function that can best reconstruct the original documents, we first need to define the loss function to evaluate the quality of the reconstruction. He et al. [11] employed the square of Euclidean distance as the loss function. Since every sentence is represented by topic distribution, we use the generalized Kullback-Leibler divergence to measure the divergence between the summary sentences and the original documents. The loss function of TopicDSDR can be defined:

$$D(v_i||f_i(X; a_i)) = \sum_{k=1}^K v_{ik} \log \frac{v_{ik}}{(f_i(X; a_i))_{ik}} - v_{ik} + (f_i(X; a_i))_{ik} \quad (7)$$

where $V = [v_1, v_2, \dots, v_N]^T \in \mathbb{R}^{N \times K}$, K is the topic number. We set $K = 10$ for TopicDSDR. The objective function of TopicDSDR can be defined as:

$$\min_{a_i} \sum_{i=1}^N D(v_i||f_i(X; a_i)) \quad (8)$$

4.3 Topic Linear Reconstruction

For the topic linear reconstruction, the optimal objective function can be defined as follow:

$$\min_{a_i} \sum_{i=1}^N D(v_i||X^T a_i) + \lambda ||a_i||_1$$

$$s.t. \quad X \subset V, |X| = M \text{ and } A = [a_1, a_2, \dots, a_N]^T \in \mathbb{R}^{N \times M} \quad (9)$$

where λ is the regular parameter.

The optimization problem in Eq.(9) is NP-hard since we need to choose M sentences from N sentences, more detail discussion can be found in [11]. However, the Topic linear reconstruction is more complicated than linear reconstruction in [11], since it is difficult to eliminate the parameter A by matrix transformation using the Transductive Experiment Design(TED) [24]. Since we only need to select a small number of sentences (e.g. 10 to 20) from the original documents, we develop a greedy select algorithm to get a sub-optimal solution. We begin with the definition of an auxiliary function [15]:

Definition 1. $G(a, a^{(t)})$ is an auxiliary function for $F(a)$ if the following conditions

$$G(a, a^{(t)}) \geq F(a), \quad G(a, a) = F(a), \quad (10)$$

are satisfied.

Lemma 1. If G is an auxiliary function of F , then F is non-increasing under the update

$$a^{(t+1)} = \operatorname{argmin} G(a, a^{(t)}) \quad (11)$$

Proof. $F(a^{(t+1)}) \leq G(a^{(t+1)}, a^{(t)}) \leq G(a^{(t)}, a^{(t)}) = F(a^{(t)})$. \square

Suppose we have already selected $M - 1$ sentences as the summary, now we want to select the M -th sentence from the rest candidate set C to form M sentences as the summary that best reconstruct the original text. We need to minimize Eq.(9) and we have the following lemma:

Lemma 2. Function

$$\begin{aligned} G(a, a^{(t)}) &= \sum_{i=1}^N \sum_{k=1}^K (v_{ik} \log v_{ik} - v_{ik} + \sum_{m=1}^M a_{im} x_{mk}) \\ &\quad - v_{ik} \sum_{m=1}^M \frac{a_{im}^{(t)} x_{mk}}{\sum_{p=1}^M a_{ip}^{(t)} x_{pk}} (\log a_{im} x_{mk} - \log \frac{a_{im}^{(t)} x_{mk}}{\sum_{p=1}^M a_{ip}^{(t)} x_{pk}}) + \lambda \|a_i\|_1 \end{aligned} \quad (12)$$

is an auxiliary function for $F(a)$:

$$\begin{aligned} F(a) &= \sum_{i=1}^N D(v_i || X^T a_i) + \lambda \|a_i\|_1 \\ &= \sum_{i=1}^N \sum_{k=1}^K (v_{ik} \log v_{ik} - v_{ik} + \sum_{m=1}^M a_{im} x_{mk}) - v_{ik} \log \sum_{m=1}^M a_{im} x_{mk} + \lambda \|a_i\|_1 \end{aligned} \quad (13)$$

Proof. It is easy to verify that $G(a, a) = F(a)$, we only need to prove $G(a, a^{(t)}) \geq F(a)$. We use the convexity of the log function

$$-\log \sum_{m=1}^M a_{im} x_{mk} \leq -\sum_{m=1}^M \pi_m \log \frac{a_{im} x_{mk}}{\pi_m} \quad (14)$$

which holds for all nonnegative π_m that sum to unity. Setting

$$\pi_m = \frac{a_{im}^{(t)} x_{mk}}{\sum_{p=1}^M a_{ip}^{(t)} x_{pk}} \quad (15)$$

then we have :

$$-v_{ik} \log \sum_{m=1}^M a_{im} x_{mk} \leq -v_{ik} \sum_{m=1}^M \pi_m (\log a_{im} x_{mk} - \log \pi_m) \quad (16)$$

From this inequality, it follows that $G(a, a^{(t)}) \geq F(a)$. \square

Now the derivative of $G(a, a^{(t)})$ with respect to a_{im} is :

$$\begin{aligned} \frac{\partial G(a, a^{(t)})}{\partial a_{im}} &= \sum_{k=1}^K (x_{mk} - v_{ik} \frac{a_{im}^{(t)} x_{mk}}{\sum_{p=1}^M a_{ip}^{(t)} x_{pk}} \frac{1}{a_{im}}) + \lambda \text{sign}(a_{im}) \\ &= 1 - \sum_{k=1}^K v_{ik} \frac{a_{im}^{(t)} x_{mk}}{\sum_{p=1}^M a_{ip}^{(t)} x_{pk}} \frac{1}{a_{im}} + \lambda \text{sign}(a_{im}) \end{aligned} \quad (17)$$

where x_m is a probability distribution and $\sum_{k=1}^K x_{mk} = 1$. $\text{sign}(\cdot)$ is the sign function.

By setting the above derivative to be zero, we get the updating formula of a_{im} :

$$a_{im} \leftarrow a_{im}^{(t)} \frac{\sum_{k=1}^K v_{ik} \frac{x_{mk}}{\sum_{p=1}^M a_{ip}^{(t)} x_{pk}}}{1 + \lambda \text{sign}(a_{im})}, \lambda \neq \pm 1, m = 1 \text{ to } M. \quad (18)$$

Algorithm 1 describes the TopicDSDR with linear reconstruction. In step(3) and (6), we select a sentence as the seed sentence that minimizes Eq.(9) with $a_{ij} = \frac{1}{N}$, which means every sentence has the equal chance to be selected as the summary. Then in the m -th iteration, for every candidate v_i in the rest of set C , we put it in position m , using above updating formula to obtain the weight a_{*v_i} . In step (15), we choose a sentence with maximum column weight score as the best candidate: $\text{score}(\text{SumColumn}(A_{*j}))$, SumColumn means the sum of column j in A . Finally, we get the summary sentences set with M sentences. The main cost of algorithm 1 is from (8) to (17). $O(NNK)$ for step (3) to (5). $O(TN MK)$ for step (12) to (16), where T is maximum iteration. Then we have $O(NNK + MN(TN MK)) = O(KN^2 + KTM^2N^2)$ overall, where K and M are relatively small.

4.4 Topic Nonnegative Linear Reconstruction

For the topic nonnegative reconstruction, we have the following objective function with regularization:

$$\begin{aligned} \min_{a_i, \beta_j} \quad & \sum_{i=1}^N \{ D(v_i || V^T a_i) + \sum_{j=1}^N \frac{a_{ij}}{\beta_j} \} + \lambda \|\beta\|_1 \\ \text{s.t. } \beta_j \geq 0, a_{ij} \geq 0 \text{ and } a_i \in \mathbb{R}^N \end{aligned} \quad (19)$$

here we use $\frac{a_{ij}}{\beta_j}$ instead of $\frac{a_{ij}^2}{\beta_j}$ which was introduced in [11]. We introduce an auxiliary function for the topic nonnegative reconstruction:

Lemma 3. *Function*

$$\begin{aligned} G(a, a^{(t)}) = & \sum_{i=1}^N \sum_{k=1}^K (v_{ik} \log v_{ik} - v_{ik} + \sum_{j=1}^N a_{ij} v_{jk}) \\ & - v_{ik} \sum_{j=1}^N \frac{a_{ij}^{(t)} v_{jk}}{\sum_{p=1}^N a_{ip}^{(t)} v_{pk}} (\log a_{ij} v_{jk} - \log \frac{a_{ij}^{(t)} v_{jk}}{\sum_{p=1}^N a_{ip}^{(t)} v_{pk}}) + \sum_{j=1}^N \frac{a_{ij}}{\beta_j} + \lambda \|\beta\|_1 \end{aligned} \quad (20)$$

is an auxiliary function for $F(a)$:

$$F(a) = \sum_{i=1}^N \{ D(v_i || V^T a_i) + \sum_{j=1}^N \frac{a_{ij}}{\beta_j} \} + \lambda \|\beta\|_1 \quad (21)$$

Similar prove can be seen in **Lemma 2**. By fixing a_i and setting the derivative of G with respect to β to be zero, we obtain the minimum solution of β :

$$\beta_j = \sqrt{\frac{\sum_{i=1}^N a_{ij}}{\lambda}} \quad (22)$$

The derivative of G with respect to A with β fixing is :

$$\frac{\partial G}{\partial a_{ij}} = - \sum_{k=1}^K v_{ik} \frac{a_{ij}^{(t)} v_{jk}}{\sum_{p=1}^N a_{ip}^{(t)} v_{pk}} \frac{1}{a_{ij}} + (1 + \frac{1}{\beta_j}) \quad (23)$$

By setting the above derivative to be zero, we get the updating formula of a_{ij} :

$$a_{ij} \leftarrow \sum_{k=1}^K v_{ik} \frac{a_{ij}^{(t)} v_{jk}}{\sum_{p=1}^N a_{ip}^{(t)} v_{pk}} \frac{\beta_j}{1 + \beta_j} \quad (24)$$

Under the above updating rule, we can find at least a locally optimal solution, more detail can found in [15]. Algorithm 2 describes the TopicDSDR with non-negative linear reconstruction. In each iteration, we first calculate the score of β_j according to Eq.(22). Then by fixing β , we update a_{ij} according to Eq.(24) until it converges. After several iterations, a majority elements of β are equal to zeros and the nonnegative values of β are chosen as the summary for the document set. Suppose the maximum number of iterations in step (4) and (6) are T_1 and T_2 respectively, the complexity of algorithm 2 is $O(T_1 N + T_2 K N^2)$.

Algorithm 1. TopicDSDR with linear reconstruction

Input:

- The candidate sentence set: $V = [v_1, \dots, v_N]^T$
- The number of sentences to be selected: M
- The regularizing parameter: $\lambda > 0$

Output:

- The select summary sentence set: $X \subseteq V$

```

1: procedure :
2: initialize  $a_{ij}$ ,  $X \leftarrow \emptyset$ ,  $C \leftarrow V$ ,  $C$  is the candidate sentence set.
3:   for  $j = 1$  to  $N$  do
4:      $score(x_j) \leftarrow \sum_{i=1}^N \sum_{k=1}^K v_{ik} \log \frac{v_{ik}}{a_{ij} x_{jk}} - v_{ik} + a_{ij} x_{jk} + \lambda ||a_i||_1$ ,  $a_{ij} = \frac{1}{N}$ 
5:   end for
6:    $x_i \leftarrow \text{argmin } score(x_j)$ 
7:    $X \leftarrow X \cup x_i$  ,  $C \leftarrow C - \{x_i\}$ 
8:   for  $m = 2$  to  $M$  do
9:     for each  $x_i \in C$  do
10:       $X\_cand \leftarrow X \cup x_i$ 
11:      repeat
12:         $a_{im} \leftarrow a_{im}^{(t)} \frac{\sum_{k=1}^K v_{ik} \frac{x_{mk}}{\sum_{p=1}^M a_{ip}^{(t)} x_{pk}}}{1 + \lambda \text{sign}(a_{im})}$  ,  $x_m \in X\_cand$ 
13:        until converge ;
14:      end for
15:       $x_i \leftarrow \text{argmax } score(\text{SumColum}(A_{*j}))$  ,  $j \in C$ 
16:       $X \leftarrow X \cup x_i$  ,  $C \leftarrow C - \{x_i\}$ 
17:   end for
18: return X;
19: end procedure

```

Algorithm 2. TopicDSDR with nonnegative linear reconstruction

Input:

- The candidate sentence set: $V = [v_1, \dots, v_N]^T$
- The regularizing parameter: $\lambda > 0$

Output:

- The select summary sentence set: $X \subseteq V$

```

1: procedure :
2: initialize  $a_{ij}, \beta_j$  ,  $X \leftarrow \emptyset$ 
3:   repeat
4:      $\beta_j = \sqrt{\frac{\sum_{i=1}^N a_{ij}}{\lambda}}$ 
5:     repeat
6:        $a_{ij} \leftarrow \sum_{k=1}^K v_{ik} \frac{a_{ij}^{(t)} v_{jk}}{\sum_{p=1}^N a_{ip}^{(t)} v_{pk}} \frac{\beta_j}{1 + \beta_j}$ 
7:     until converge
8:   until converge
9:    $X \leftarrow \{v_i | v_i \in V, \beta_i \neq 0\};$ 
10: return X;
11: end procedure

```

Table 1. Comparison results on DUC2006

Methods	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SU
LDA-based	0.34187	0.05506	0.28045	0.10806
LDA-SVD	0.35254	0.05339	0.28266	0.10975
NMF	0.35789	0.06489	0.30046	0.12424
LinDSDR-Word	0.356	0.06225	0.32673	0.11699
NonDSDR-Word	0.30207	0.03579	0.27503	0.08903
LinDSDR-Prob	0.35205	0.06123	0.32488	0.11532
NonDSDR-Prob	0.358	0.06233	0.32949	0.11736
LinTopicDSDR	0.32954	0.04696	0.30553	0.10158
NonTopicDSDR	0.37635	0.07073	0.34172	0.1319

5 Experiments and Analysis

5.1 Data Sets and Evaluation System

To evaluate our multi-document summarization algorithms, we use the DUC2006 and DUC2007 data sets from Document Understanding Conference(DUC) for generic automatic summarization evaluation. DUC2006 and DUC2007 have 50 and 45 document sets respectively, each with 25 news articles.

We use the ROUGE(Recall-Oriented Understudy for Gisting Evaluation) toolkit[17] to evaluate the proposed methods, which has been widely applied by DUC for performance evaluation. It measures the quality of a summary by counting the unit overlaps between the candidate summary and a set of references summaries. In this experiment results, we show four F measures of ROUGE metrics: ROUGE-1(unigram-based), ROUGE-2(bigram-based), ROUGE-SU4(the skip-bigram co-occurrences statistics) and ROUGE-L(the longest common sub-sequence).

5.2 Implemented Systems and Result Analysis

To compare with our proposed algorithms, we implement the following document summarization methods as the baseline systems.(1)LDA-based [1]. (2)LDA-SVD [2]. (3)NMF [16]. Methods (1)-(3) can be seen in section 2 for details. (4)DSDR: the original DSDR using the Euclidean distance proposed in [11]. We evaluate it on two feature spaces. The first is word space: LinDSDR-Word and NonDSDR-Word. The second is topic probability space: LinDSDR-Prob and NonDSDR-Prob. (5)TopicDSDR. Our TopicDSDR using the KL divergence are named LinTopicDSDR and NonTopicDSDR respectively.

Table 1 and Table 2 show the experiment results between TopicDSDR and other methods. From the results, we have the following observations:

(1) The NonDSDR-Word based on term frequency has the worst performance, since the number of summary sentences generated by NonDSDR-Word is relatively large and the final β_j values are almost the same for each candidate

Table 2. Comparison results on DUC2007

Methods	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-SU
LDA-based	0.36518	0.06208	0.29004	0.1194
LDA-SVD	0.3738	0.06941	0.30265	0.12572
NMF	0.37355	0.07192	0.30891	0.13365
LinDSDR-Word	0.36796	0.07034	0.33585	0.12411
NonDSDR-Word	0.31754	0.04092	0.28547	0.09563
LinDSDR-Prob	0.3697	0.06884	0.33941	0.12259
NonDSDR-Prob	0.37215	0.07198	0.34011	0.12613
LinTopicDSDR	0.34652	0.05654	0.31887	0.11217
NonTopicDSDR	0.39849	0.082	0.36164	0.14562

Table 3. Average summary sentence numbers of nonnegative reconstruction

Corpus	NonDSDR-Word	NonDSDR-Prob	NonTopicDSDR
DUC2007	559.16	546.82	17.33
DUC2006	714.22	700.16	16.5

summary sentence. (2) The score of LinDSDR-Prob and NonDSDR-Prob are very closed. The LinDSDR-Prob does not have the weight parameter A to be learned. It takes few seconds to generate a summary for a document set while the NonDSDR-Prob may take half an hour. While our LinTopicDSDR is relatively poor since we need to learn the parameter A . (3) Our NonTopicDSDR outperforms the original DSDR and the other implemented systems. This is because the extracted summary based on word space would be dominated by the frequency of the words. The topic decomposition can distinguish words from different topics. What's more, our model reduces word space to topic space which is more efficient. The number of sentences generated by nonnegative DSDR and our TopicDSDR can be seen in Table 3, which shows that our approach is more closed to the natural form of summary. (4) The NMF outperforms the LDA-SVD and LDA-based. The NMF with nonnegative weight and feature vectors, which is similar to the nonnegative reconstruction.

6 Conclusion and Future Work

In this paper, we propose a novel summarization model by incorporating the topic decomposition with data reconstruction, named TopicDSDR. We develop an optimization frame work based on KL divergence. The experiment results demonstrated TopicDSDR can not only best reconstruct the original documents but also capture the semantics similarity, since it directly optimizing the topic space rather than the word space. TopicDSDR is more efficient than the original DSDR model and more closed to the natural form of summary. It is interesting

to develop effective solutions for linear TopicDSDR by eliminating the parameter A in our future work.

Acknowledgments. We thank the anonymous reviewers for their valuable and constructive comments. This work is financially supported by NSFC Grant 61272340.

References

1. Arora, R., Ravindran, B.: Latent dirichlet allocation based multi-document summarization. In: AND, pp. 91–97 (2008)
2. Arora, R., Ravindran, B.: Latent Dirichlet Allocation and Singular Value Decomposition Based Multi-document Summarization. In: ICDM, pp. 713–718 (2008)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. JMLR, 993–1022 (2003)
4. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks, 107–117 (1998)
5. Cai, D., He, X., Han, J., Huang, T.S.: Graph Regularized Nonnegative Matrix Factorization for Data Representation. IEEE Trans. Pattern Anal. Mach. Intell., 1548–1560 (2011)
6. Carbonell, J.G., Goldstein, J.: The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In: SIGIR, pp. 335–336 (1998)
7. Edmundson, H.P.: New methods in automatic extracting. Journal of the ACM 16(2), 264–285 (1969)
8. Erkan, G., Radev, D.R.: LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. J. Artif. Intell. Res. (JAIR), 457–479 (2004)
9. Gong, Y., Liu, X.: Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In: SIGIR, pp. 19–25 (2001)
10. Haghghi, A., Vanderwende, L.: Exploring Content Models for Multi-Document Summarization. In: HLT-NAACL, pp. 362–370 (2009)
11. He, Z., Chen, C., Bu, J., Wang, C., Zhang, L., Cai, D., He, X.: Document Summarization Based on Data Reconstruction. In: AAAI (2012)
12. Hennig, L.: Topic-based Multi-Document Summarization with Probabilistic Latent Semantic Analysis. In: International Conference RANLP, Borovets, Bulgaria, pp. 144–149 (2009)
13. Hofmann, T.: Probabilistic Latent Semantic Analysis. In: UAI, pp. 289–296 (1999)
14. Kleinberg, J.M.: Authoritative Sources in a Hyperlinked Environment. In: SODA, pp. 668–677 (1998)
15. Lee, D.D., Seung, H.S.: Algorithms for Non-negative Matrix Factorization. In: NIPS, pp. 556–562 (2000)
16. Lee, J., Park, S., Ahn, C., Kim, D.: Automatic generic document summarization based on non-negative matrix factorization. Inf. Process. Manage., 20–34 (2009)
17. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: ACL Workshop, pp. 74–81 (2004)
18. Lin, H., Bilmes, J.: Multi-document Summarization via Budgeted Maximization of Submodular Functions. In: HLT-NAACL, pp. 912–920 (2010)
19. Luhn, H.P.: The automatic creation of literature abstracts. IBM JRD 2(2), 159–165 (1958)

20. McDonald, R.: A study of global inference algorithms in multi-document summarization. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECIR 2007. LNCS, vol. 4425, pp. 557–564. Springer, Heidelberg (2007)
21. Mihalcea, R., Tarau, P.: Text-rank: bringing order into texts. In: EMNLP, pp. 404–411 (2004)
22. Woodsend, K., Lapata, M.: Multiple Aspect Summarization Using Integer Linear Programming. In: EMNLP-CoNLL, pp. 233–243 (2012)
23. Phan, X.-H., Nguyen, C.-T.: GibbsLDA++: A C/C++ implementation of latent Dirichlet allocation, LDA (2007)
24. Yu, K., Bi, J., Tresp, V.: Active learning via transductive experimental design. In: ICML, pp. 1081–1088 (2006)

CUVIM: Extracting Fresh Information from Social Network

Rui Guo, Hongzhi Wang, Kaiyu Li, Jianzhong Li, and Hong Gao

Harbin Institute of Technology

{ruiguo,wangzh,likaiyu,lijzh,honggao}@hit.edu.cn

Abstract. Social network preserves the life of users and provides great potential for journalists, sociologists and business analysts. Crawling data from social network is a basic step for social network information analysis and processing. As the network becomes huge and information on the network updates faster than web pages, crawling is more difficult because of the limitations of bandwidth, politeness etiquette and computation power. To extract fresh information from social network efficiently and effectively, this paper presents a novel crawling method of social network. To discover the feature of social network, we gather data from real social network, analyze them and build a model to describe the discipline of users' behavior. With the modeled behavior, we propose methods to predict users' behavior. According to the prediction, we schedule our crawler more reasonably and extract more fresh information. Experimental results demonstrate that our strategies could obtain information from SNS efficiently and effectively.

Keywords: Social Network, Crawler, Freshness.

1 Introduction

1.1 Motivation

Social Network Service is one of the hottest services in the last few years. It has a tremendous group of users. For instance, Facebook has 800 million active users [1] and Twitter reaches 500 million users. It is estimated that at least 2.3 billion tweets have been published on Twitter during a 7-month period, more than 300 million tweets per month [2]. And The Yahoo! Firehouse has reached 750K ratings per day, 150K comments per day [3].

There are a few social media datasets such as Spinn3r [4]. About 30 million articles (50GB of data), including 20,000 news sources and millions blogs were added to spinne3r per day [2]. As people access SNS frequently, advertisement could be broadcast according to users' behavior. [16] studies various Super Bowl ads by applying data mining techniques through Twitter messages. Using Twitter, [17] detects earth quakes and [18] studies influenza spread.

For crawling, one of the most important factors is the freshness. Denev, Mazeika, Spaniol and Weikum [5] designed a web crawling framework named SHARC. It pays more attention on the relationship between freshness and time period. Moreover,

Olston and Pandey proposed a crawling strategy optimized for freshness [6], it concerns more about the current time point. Even though SNS crawling is related to web crawling, crawling SNS for fresh information is different from web crawling in following points and brings new technical challenges.

1. New messages are published more frequently.
2. The messages on SNS are shorter than web pages.
3. The SNS is closely related to users' daily life.
4. SNS network is more complex.

With these features, new techniques for crawling fresh SNS information are in demand. With the consideration that the goal of crawling SNS information is to gather new information, this paper aims to crawl as more new messages as possible with the limited resources.

1.2 Contributions

For crawling, the limitations on resources include bandwidth, computation power and politeness etiquette. For instance, for *twitter.com*, we may be permitted to get at most 200 tweets with a Twitter API call. The restriction on the method call is 350 calls per hour for one authorized developer account [23]. Actually, the API restriction is the bottleneck of most SNS crawlers.

To leverage the limited resources and freshness requirements of the crawler, we classify the users according to their behaviors and model their behavior of updating posts respectively. With these models, the time of post updating for different users is predicted and the crawler could access the posts of users only when corresponding is updated. As a result, the latest information is collected with limited resources.

Combing the steps discussed above, we propose Crawl based on User Visiting Model (CUVIM in brief) based on the observations and classifications of users' behaviors. In this paper, we focus on the messages in SNS. Considering the relationship updating between users as a special type of message, the relationship of updating information could also be crawled with the techniques in this paper.

According to different behaviors on updating the posts, we classify SNS users into 4 kinds: the inactive account, frequently changing account, reasonable constant account, and authority account. This is the first contribution of this paper.

We design different updating time predication model and accordingly develop efficient crawling strategies. This is the second contribution of this paper.

Concretely, for the inactive accounts and frequently changing accounts which change not very frequently, the changes can be described by the Poisson process and the changing rate can be predicted by statistics methods. Thus we build the Poisson model and take web crawling strategy to crawl SNS data.

From reasonable constant accounts and authority accounts who post messages frequently, we observe that the frequency of new messages is related to users' daily lives. According to this observation, we can crawl many fresh and useful messages in the day while almost no new messages at night. And we build the Hash Model to visit those active users to crawl the information efficiently.

As the third contribution, extensive experiments are performed to verify the effectiveness and efficiency of the techniques in this paper. We crawl 10,000 users and collect 1,853,085 messages that they post during 2012-11-01 and 2013-01-01. From

experimental results, the Poisson Process Model collects 12.14% more messages than a round-and-robin (RR) style method. The Hash Model collects about 50% more messages than a RR style method.

This paper is organized as follows. Section 2 reviews related work at crawler field; Section 3 introduces our work and discovery from the data; Section 4 introduces our crawl altimeters; Section 5 shows the experimental results. Section 6 draws the conclusions and proposes further research problems.

2 Related Work

Only a few methods are proposed to crawl SNS data.

[16] describes a Twitter Crawler developed by Java. They pay more attention to the implementation detail of the crawler and the data analysis. Instead, we focus on the crawling method and develop algorithms to gather more information of the specific SNS users.

TwitterEcho is an open source Twitter crawler developed by Bošnjak et al. [19]. It applied a centralized distributed architecture. Cloud computing is also used for SNS crawler [20], Noordihuis et al. collect Twitter data and rank Twitter users through the PageRank algorithm. Another attempt is to crawl parallel. Duen et al. implemented a parallel eBay crawler in Java and visited 11,716,588 users in 23 days [22]. The three methods aim to get more calculating resource, while we focus on a more reasonable crawling sequence with the given resource.

Whitelist accounts were once available on Twitter. Kwak et al. crawled the entire Twitter site successfully, including 41.7 million user profiles and 106 million tweets by Twitter API [21]. However, whitelist accounts are no longer available now. It is the same for [20]. As the API has rate-limiting now, we propose algorithms to improve the crawl efficiency.

Another related work to SNS message collector is web crawlers. Generally the page changing follows the Poisson process model. The Poisson process model assumes that the changing rate λ_i in every same time unit Δ is the same. And the changes are modeled as following formula.

$$P[\text{changes in } \Delta \text{ is } k] = \frac{e^{-\lambda_i \Delta} (\Delta \lambda_i)^k}{k!}.$$

This equation can predict the possible changes of web pages and can be applied on only the inactive SNS users, since they usually change constantly. However, when considering the active SNS users who post messages frequently, the change rate is not equal all the time. The discipline of SNS messages follows the users' daily life. A user may post many messages in the day and much less at night, so the changing rate is not same for the day and night.

Many web crawlers are proposed. Reprehensive measures for web crawling are sharpness [5] and freshness [6]. The strategies define sharpness or freshness to the crawling, and schedule crawling to achieve those targets. Differently, we choose the total number of new SNS messages as our target, and schedule according to the SNS users' behavior.

There are other matured web crawling strategies. J. Cho, H. Garcia-Molina and L. Page improve the crawling efficiency through URL ordering [11]. They define several

importance metrics including ordering schemes and performance evaluation measures to obtain more “important” URL first. J. Cho, H. Garcia-Molina also proposes a strategy for estimating the change frequency of pages to make the web crawler work better [12] by identifying scenarios and then develop “frequency estimators”. C. Castillo, M. Marin, A. Rodriguez and R. Baeza-Yates combine the breadth-first ordering with the largest sites first to crawl pages fast and simple [13]. J. Cho and U. Schonfeld make pagerank coverage guarantee high personalized to improve the crawler [14].

3 SNS Data Analysis and Classification

At first, to design the proper crawler for SNS, we discuss the features of SNS data and the classification of SNS users by their behavior in this section. Considering the four features of the SNS in Section 1, we propose novel methods. At first, we define audience and channel as following in SNS relationships to study the SNS, where both A and B are users.

Definition 1 (audience). Audience is a one way SNS relationship on SNS. A is B ’s audience, that means, A can check B ’s SNS messages.

Definition 2 (channel). Channel is a one way SNS relationship on SNS. A is B ’s channel, that means, A ’s messages will be checked by B .

To study the behavior of SNS users of messages updating, we crawl and study SNS messages. For effective crawling, we choose several top SNS users from the influence ranking list in Sina weibo (<http://weibo.com>), which is a famous SNS with millions of users. They are added to the crawling list as the seeds and some of their channels are randomly selected. The channels behaves more active than the audiences, thus we can avoid invalid accounts in the list. The channels are added to the crawling list. Then they are treated as the new seeds and their channels are accessed. We can end the iterations when we get enough users in the crawling list.

With a seeds, k channels are chosen and n -hop channels for each seed are traversed, the crawling list contains $\sum_{i=0}^n ak^n = a(k^{n+1} - 1)/(k - 1)$ users. From this formula, the smaller n is, the more representative the users in the list are: at the beginning the seed users are famous, and later more and more ordinary people are added into the list. Since without any information, it is impossible to predict the frequency of posting, initially, we have crawled the data for all users in a round-and-robin style.

We crawled 10,000 users’ data for two months, and got 1,853,085 messages in total. According to the experimental results, we have following observations.

1. Users are quite different in posting frequency. The users in the crawling list have at least one audience. It means that those accounts are valid and are or were once active. However, during the experiment period, about 1/2 users post less than one message per day. And 1/5 users post more than 10 messages per day. This observation means that the crawl frequency for different users should be different.
2. The frequency of new messages may change with time. An extreme user post 18 messages in the first day but did not post anything in the following three days.
3. The frequency of posting new messages is related to the users’ daily life. Experimental result shows that an actress posts about half messages in the late night while a manager posts most messages in the afternoon.

4. Some accounts are maintained by professional clerks or robots, such as a newspaper's official account. Those accounts have more audiences, change more frequently and have more influence than the personal users.

With above observations, we classify all users into 4 types by their behavior. To illustrate the features of these four kinds of users, we show the experimental results of four users belonging to each type in Figure 1 to 4. It shows the total number of new messages in each 15 minutes of a day. The horizontal axis means time from 0:00 to 24:00 in a day, and the longitudinal axis means the total number of new messages post with 15 minutes as the unit. In Figure 1, 3 and 4, the line in the figure means the number of all messages in the 2 months. In Figure 2, each line means a day.

1. *Inactive account*. The users in this type post nothing in a long period or have little channels and audiences.

From Figure 1, it can be observed that the figure of inactive account has at most a few points. It means that the user has hardly post any messages in a day, but may a few messages in a few weeks. When observing for a long period, those users behave as web pages. It means that an inactive user may post three messages a month while a page may change three times this month. The number of possible changes between each equal time unity Δ when Δ is large enough. Thus we can descript the behavior of this type by Poisson process.

2. *Instable changing account*. Such type of users' behavior is instable and cannot be predicted. Users in this class are often very active at a short period and remain silence later. For example, the one who post 18 messages and did not post anything in later 3 days. It is hard to design an effective crawling strategy for those people.

Figure 2 shows the instable changing account. The user post 2 messages on Monday, 13 messages on Tuesday and nothing in the later three days. It is the most irregular one among all figures. It may be illustrated that the user has a sudden trip and cannot connect to the SNS, or the work those days are busy so he pays little attention to SNS. And those users may become reasonable constant users when he returns or finishes the work. There is no effective strategy to crawl this kind of users, for we cannot predict their behavior, thus we cannot schedule the crawler well. We treat those users as the inactive accounts to save crawling resource, and put them into the reasonable constant users once. We found that they post many messages every day in the recent week.

3. *Reasonable constant user*. Most valid accounts belong to this type. For example, the users with work far from computer or mobile phone have to post messages after work and the users working with computer will post in work day in the office. The frequency of the new messages is obviously influenced by the users' daily life, and the new messages often occur frequently in the afternoon and at night when the user takes a rest. Hence we can predict their behavior by historic data.

Figure 3 shows the behavior of reasonable constant user. Such users love the SNS very much and post messages very regularly. This kind of figures often has two peaks, the afternoon and the night. The curve in the left of the peak grows up and the one after the peak goes down.

4. *Authority account*: Such accounts are maintained by several clerks or just robots. The content of those accounts is carefully updated far more frequent than ordinary people and is reviewed by more users. For example, the *New York Time* updates news quickly on Twitter and has a large group of audience.

Figure 4 shows the behavior of authority account. The kind of figures is higher than the usual users' and has more peaks than the reasonable constant use. From the result, there are one peak for each hour, and the peaks are almost the same high. The reason is that clerks or robots may be asked to post messages once an hour.

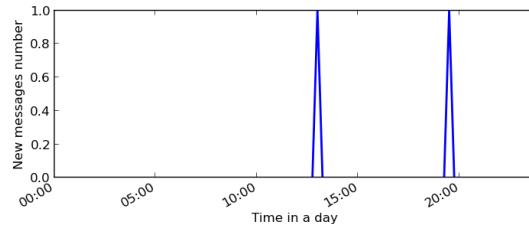


Fig. 1. The Updating Ratio of an Inactive Account

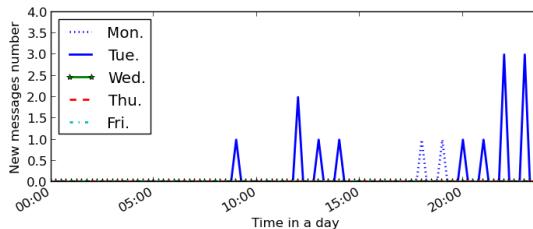


Fig. 2. The Updating Ratio of an Inactive Account

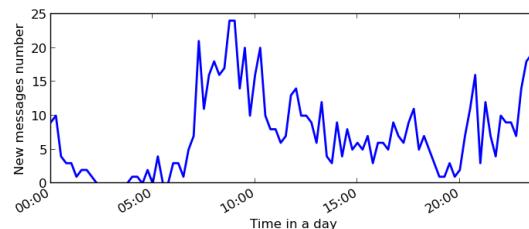


Fig. 3. The Updating Ratio of an Inactive Account

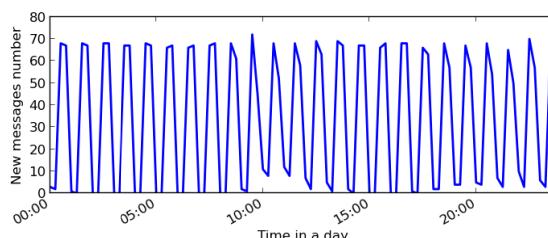


Fig. 4. The Updating Ratio of an Authority account

4 User Behavior Model and Crawl Strategy

According to the behavior of the users, we build two models for effective crawling, the Poisson Model and Hash Model.

The Poisson Model is built for inactive users, who behave similar to web pages, and the effective web crawling strategy such as SHARC [5], trade-offs [8] and sampling [9] can be applied to those users.

The Hash model records the discipline of very active users' behavior. For those users, we consider the frequency change in a day, and the change rate λ_i is not same all the day long. Hence the Poisson process cannot descript it. One effective way is to record the historic data change rate to predict the parameters for crawling.

4.1 Poisson Process Model

According to the observation of inactive users, we build a Poisson Process Model for their posting frequency, since those users behave like the updating of web pages, including changing constantly in a comparative long period (e.g. a month). We assume that the change rate λ_i can be estimated according to historic data, the number of audiences and channels, just as estimating change rate of pages by types, directory depths, and URL names.

However, the web crawling metrics require to be modified to fit SNS. The SHARC define the blur for web pages, it descripts the difference between the new and old pages of the same URL. Yet this could not be applied to SNS. The blur means the

possibility that a page changes, but for the SNS, the question is the possible number of new messages, rather than whether a new message is post or not. Thus the blur cannot descripts the messages. To describe the freshness of messages in SNS, we define the potentiality for the users and the crawling target is to minimize the total potentiality. We define the potentiality for SNS as follows.

Definition 3 (Potentiality). Let u_i be a SNS user crawled at time t_i , The potentiality of the user is the expected number of new messages between t_i and query time t , averaged through observation interval $[0, n\Delta]$:

$$P(u_i, t_i, n, \Delta) = \frac{1}{n\Delta} \int_0^{n\Delta} \lambda_i \cdot |t - t_i| dt = \frac{\lambda_i \omega(t_i, n, \Delta)}{n\Delta},$$

where

$$\omega(t_i, n, \Delta) = t_i^2 - t_i n\Delta + \frac{(n\Delta)^2}{2}$$

is the crawling penalty.

Let $U = (u_0, \dots, u_n)$ be SNS users crawled at the corresponding times $T = (t_0, \dots, t_n)$. The blur of the crawling is the sum of the potentiality of the individual users is defined as,

$$P(U, T, n, \Delta) = \frac{1}{n\Delta} \sum_{i=0}^n \lambda_i \omega(t_i, n, \Delta).$$

$\omega(t_i, n, \Delta)$ denotes the crawling penalty. It depends on the crawling time t_i and the period of the crawling interval $n\Delta$, but not on the user. And we obtain Theorem 1.

Theorem 1 (Properties of the Schedule Penalty). Double crawling delay will lead to fourfold crawling penalty and double potentiality:

$$\begin{aligned}\omega(i\Delta, n, \Delta) &= \Delta^2 \omega(i, n, 1) \\ P(U, T, n, \Delta) &= \Delta P(U, T, n, 1)\end{aligned}$$

For the interests of space, we omit the proof of the Theorem.

For instance, the change rate for user u_0 to u_5 is $\lambda_0=0$, $\lambda_1=1$, $\lambda_2=2$, $\lambda_3=3$, $\lambda_4=4$ and $\lambda_5=0$, then we should crawl those users in the order $u_0, u_2, u_4, u_5, u_3, u_1$, thus we get the potentiality minimum.

Algorithm 1 depicts the Process Model algorithm for inactive users in SNS. All the users are known advance. We can sort and scan all the users only once to schedule the crawler. The time and space complexity are both $O(n)$. Thus we scan the user list only once.

Algorithm 1. Crawl Schedule with Poisson Model

```

input : sorted users  $u_0, \dots, u_n$ 
output: crawl schedule  $u_0^D, \dots, u_n^D$ 
    For  $i = 0, 1, \dots, n$  do
        if  $i$  is even then  $u_i^D = u_{i/2}$ 
        else  $u_i^D = u_{n-(i-1)/2}$ 
```

4.2 Hash Model

According to the observation of reasonable constant users and authority accounts, we build the Hash model. The changing rate λ_i for those users is stable from the observations for a long time (e.g. a week or longer). However, the new SNS messages are posted so frequently that they need to be crawled very frequently. If we visit these users according to the averaged λ_i of the day, we will waste much resource.

For example, we visit the user in the Figure 3 according to the Poisson Process Model. In the Poisson process, the number of possible changes in each time unit is the same, so the time span between each extraction should be the same. If we start crawling at 0:00 and crawl twice a day, it may be 0:00 and 12:00. However, crawling at 3:00 and 19:00 seems the best strategy. Thus the Poisson Process Model does not fit those active users perfectly.

The number of active users' new messages changes frequently and is comparative randomly. Hence it is hard to find a precise and suitable model. On the other hand, the number of such kind of users is not large enough and a statistics model such as Gaussian model cannot be constructed according to the behaviors. One effective way to predict the users' behavior is to record the historic data.

With such considerations, we define Hash Model to obtain the messages of the active users better, including the reasonable constant users and authority accounts. This model is built for the users who need to be crawled frequently, at least twice or

more a day. This model uses a hash table to record the number of new messages in a short recent period and the earlier data has less weight. According to this historic data, we can calculate the possible number of new messages that are post in a given time span, so we can schedule the crawler better.

For example, we maintain an array of 24 bytes (a_1, \dots, a_{24}) to record the number of new messages posted by the same user in each hour. a_i values 0 at the beginning and the new value

$$a'_i = a_i * 0.5 + n * 0.5$$

where the parameter n is the number of messages posted by the user at the i th hour of the day. The weight for k days before is 0.5^k . As a result, the parameter for today is 0.5 and 0.25 for yesterday.

If a user does not post any messages for a few days, the values of the hash table decreases very fast. To avoid such phenomenon, the longest crawling span threshold s is required, that means, we will crawl the user at least once s hours or days.

Although the hash-based method can be used for active user crawling, it is not suitable for some special cases. For some special dates, such as the weekend, people behave differently from workday. The experiment results of 100 users in 2 weeks show that 1496 messages were post on Tuesday while only 874 messages on Sunday. Thus we could predict users' behavior by the last weekend data according to this feature. Similar predictions can be applied for the public holiday, such as the national day. It is required to consider about the near and similar holiday data, or even the last year's vocation data with the hash model.

Algorithm 2 depicts the Hash Model for one user.

Algorithm 2. Crawl Time Determination with Hash Model

input : $a_1, \dots, a_k, n_1, \dots, n_k$

output: crawl time list L

$lastCrawlTime = 0, sum_0 = 0 - remainingMessage$

For $i = 1, \dots, k$ *do*

$a_i = a_i * 0.5 + n_i * 0.5$

$sum_1 = a_1$

For $i = 2, \dots, k$ *do*

$sum_i = sum_{i-1} + a_i$

for $i = 1, \dots, k$ *do*

if $sum_i - sum_{lastCrawlTime} > c$ *or* $i - lastCrawlTime > s$

$L \triangleright add \leftarrow i \leftarrow$

$lastCrawlTime = i$

$remainingMessage = sum_k - sum_{lastCrawlTime}$

The length of the hash table is k , we crawl c messages each time and the user post n_i messages yesterday at the time span i , $remainingMessage$ is the number of messages that are not crawled the day before, the sum_i means the sum from a_1 to a_i .

and sum_0 is the number of minus $remainingMessage$, thus number of the remaining messages that are not crawled yesterday will be count., and the longest crawling span threshold is s . We input $a_1, \dots, a_k, n_1, \dots, n_k$ and it will output the crawl time list L .

First, we update the $lastCrawlTime$ and sum_0 , and calculate the values of the hash table and the value of $sum_1 \dots sum_n$. Second, we scan the sums. If there are enough messages to crawl ($sum_i - sum_{lastcrawltimes} > c$) or the crawl time span exceed the threshold ($i - lastcrawltimes > s$), then we add the time point i to the crawl time list L and update the lastcrawltimes.

The time and space complexity is $O(n)$.

5 Experimental Evaluation

To verify the effectiveness and efficiency of our strategies, we perform experiments in this section. The experiments are performed on a PC with 2.10GHz intel CPU and 4G RAM. We crawl the *weibo.com*, which is one of the hottest SNS in China. The weibo has 368 million registered users in 2012.08 and more than 100 million messages are post everyday [15].

We crawled 10,000 randomly selected users in 2 months days from 2012.11.01 to 2013.01.01. The result shows that 1,853,085 messages are collected, and a user post about 3.04 messages per day.

5.1 Experimental Results for Poisson Process Model

To test the effectiveness of Poisson Process Model, we crawled the 10,000 users, and accessed every user once. From observations, the Poisson Process Model crawled 421,722 messages, while the round-and-robin style crawled 376,053 messages. Thus the Poisson Process Model is 12.14% better.

5.2 Hash Model

We crawl the 10,000 users by hash model. We assume the crawling limit is 100 messages at one crawling, the longest crawling span threshold s is 30 days, and the weight for the last one day (in the example, it is 0.5) ranges from 0.4 to 0.9 step by 0.1. The result is shown in Table 1. For comparisons, we also conduct the experiment for RR, and crawl one user 2 to 5 times during the experiment period. The result is described in Table 2.

Table 1. The Hash Model Experiment Results

Weight	Message Number	Total Crawl Time	Avg #Msg. with one crawling
0.9	1451435	50421	28.79
0.8	1417908	44605	31.79
0.7	1368783	39446	34.70
0.6	1323012	35421	37.35
0.5	1255509	32211	38.98
0.4	1175464	29822	39.42

Table 2. The RR Style Method Experiment Results

One User Crawl Time	Total Crawl Time	Message Number	Avg #Msg with one crawling
5	50005	939964	18.80
4	40004	818039	20.45
3	30003	652641	21.75
2	20002	411086	20.55

The data in Table 1 shows that with the increase of the weight, the total crawl time increase and more messages will be crawled, while the crawling efficiency decreases. Thus the weight can be considered as a parameter to adjust the crawling efficiency and the limited crawling resource such as the API restriction. The data in Table 2 shows that the crawling efficiency for the RR style method is reasonable stable when the crawling frequency is low.

6 Conclusion and Future Work

To use the information in SNS effectively, it is necessary to obtain fresh SNS information. However, SNS crawler is quite different from web crawlers for the change rate is faster and the requirements for the latest messages are much more intensive. Therefore, the traditional web crawl method cannot be applied for SNS information crawling. To obtain fresh information in SNS with resource constraint, we classify users according to their behaviors. Their behaviors are modeled and crawling algorithms are proposed according to the models. Experimental results show that the Hash Model is about 50% better than the Round-Rabin method, and the Poisson Process Model is 12.14% better than the RR method with randomly selected users.

There are some possible future research directions. One is to crawl with different weight for different users, for the celebrities are more influential. How to define the weights for users and crawl optimally remains a challenge. Another direction is to obtain the hottest messages as early as possible. Study of information transmit is required so that we can predict the hot pot in SNS.

Acknowledgements. This paper was partially supported by NGFR 973 grant 2012CB316200, NSFC grant 61003046 and NGFR 863 grant 2012AA011004. Doctoral Fund of Ministry of Education of China (No.20102302120054), the Fundamental Research Funds for the Central Universities(No. HIT. NSRIF. 2013064).

References

- [1] Facebook, <http://www.facebook.com/press/info.php?statistics>
- [2] Stanford Graph Set, <http://snap.stanford.edu/data/>
- [3] Leskovec, J.: Social Media Analytics. SIGKDD, tutorial (2011)
- [4] Spinn3r, <http://www.icwsm.org/data/>
- [5] Denev, D., Mazeika, A., Spaniol, M., Weikum, G.: SHARC: Framework for Quality-Conscious Web Archiving. In: VLDB (2009)

- [6] Olston, C., Pandey, S.: Recrawl scheduling based on information longevity. In: WWW, pp. 437–446 (2008)
- [7] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 1–38 (1977)
- [8] Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachoura, V., Silvestri, F.: Design trade-offs for search engine caching. *ACM Trans. Web* 2(4), 1–28 (2008)
- [9] Cho, J., Ntoulas, A.: Efective change detection using sampling. In: VLDB, pp. 514–525 (2002)
- [10] Casella, G., Berger, R. (eds.): *Statistical Inference*. Brooks/Cole (2008)
- [11] Cho, J., Garcia-Molina, H., Page, L.: Efficient crawling through url ordering. In: WWW, pp. 161–172 (1998)
- [12] Cho, J., Garcia-Molina, H.: Estimating frequency of change. *Trans. Inter. Tech.* 3(3), 256–290 (2003)
- [13] Castillo, C., Marin, M., Rodriguez, A., Baeza-Yates, R.: Scheduling algorithms for web crawling. In: WebMedia, pp. 10–17 (2004)
- [14] Cho, J., Schonfeld, U.: Rankmass crawler: a crawler with highpersonalized pagerank coverage guarantee. In: VLDB, pp. 375–386 (2007)
- [15] Wikipedia, <http://zh.wikipedia.org/wiki/%E6%96%B0%E6%B5%AA%E5%BE%AE%E5%8D%9A>
- [16] Byun, C., Lee, H., Kim, Y.: Automated Twitter Data Collecting Tool for Data Mining in Social Network. In: RACS (2012)
- [17] Okazaki, T.M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proc. of Conf. on World Wide Web, WWW (2010)
- [18] Aramaki, E., Maskawa, S., Morita, M.: Twitter Catches, The Flu: Detecting Influenza Epidemics using Twitter. In: Proceedings of the 2011 Conference on Empirical Methods, in Natural Language Processing, Edinburgh, Scotland, UK, July 27-31, pp. 1568–1576. Association for Computational Linguistics (2011)
- [19] Bošnjak, M., Oliveira, E., Martins, J., Mendes, E., Sarmento, L.: TwitterEcho - A Distributed Focused Crawler to Support Open Research with Twitter Data. In: WWW 2012 – MSND 2012 Workshop, Lyon, France, April 16-20 (2012)
- [20] Noordhuis, P., Heijkoop, M., Lazovik, A.: Mining Twitter in the Cloud. In: IEEE 3rd International Conference on Cloud Computing (2010)
- [21] Dziczkowski, G., Bougueroua, L., Wegrzyn-Wolska, K.: Social Network – An tutoonous system designed for radio recommendation. In: International Conference on Computational Aspects of Social Networks, SASoN (2009)
- [22] Chau, D., Pandit, S., Wang, S., Faloutsos, C.: Parallel Crawling for Online Social Networks. In: WWW (2007)
- [23] Twitter Rate Limiting, <https://dev.twitter.com/docs/rate-limiting>

Comments-Oriented Document Summarization Based on Multi-aspect Co-feedback Ranking

Lifu Huang, Hongjie Li, and Lian'en Huang^{*}

Shenzhen Key Lab for Cloud Computing Technology and Applications
Peking University Shenzhen Graduate School
Shenzhen, Guangdong 518055, P.R. China
`{warrior.fu,lihongjie99}@gmail.com, hle@net.pku.edu.cn`

Abstract. With the popularity of Web 2.0, comments left by readers on web documents have drawn much attention. In this paper, we study the problem of comments-oriented document summarization, which aims to summarize a web document by considering not only its content but also the comments. Generally, most of the comments usually convey one or a few aspects of the document. Given a sentence set from both the web document and its corresponding comments to summarize, we can divide different sentences into different clusters (named “aspects”) according to the content. It is challenging and interesting to summarize the web document based on these clusters. Motivated by this, we propose a novel model: MultiAspectCoRank, for comments-oriented document summarization. Firstly we rank all the sentences based on the multiple aspects obtained from the whole document, and then provide each ranking list as feedback to others until the top-N results of each ranking list are unchanged. We get the final result by integrating these different ranking lists together. Experimental results on a set of real-world blog data with manually labeled sentences show the promising performance of our approach.

Keywords: Comments-Oriented Summarization, Co-Feedback Ranking, Affinity Propagation.

1 Introduction

Automatic Document summarization (ADS) has been widely studied in past decades. It aims to generate a compressed summary by extracting the major information from a document. Existing methods mostly focus on the document’s content information while ignoring readers’ opinions. With the development of web 2.0, readers are inspired to express and share their views by commenting. This kind of information coming from readers is much valuable for some special IR (Information Retrieval) tasks. On the one hand, these comments are generated based on readers’ understanding of the document, which to a certain extent reflect some aspects of the document. On the other hand, comments, which have

* Corresponding author.

consistency with document text on the content, can be viewed as an additional part of the document. In this paper, we focus on comments-oriented document summarization (CODS), aiming to generate summaries for a single web document based on not only its content but also the comments generated by its readers.

The research of comments-oriented document summarization (CODS) is popular in recent years. Its task is to summarize a document by extracting representative sentences from the document based on the information hidden in its comments [3]. There are two common challenges for comments-oriented document summarization: the first is that a comment set might contain diverse information, which is either related or unrelated to the document content, so it is necessary to design effective methods to extract useful information from comments to help summarize document content; the second is that based on the information extracted from comments, how to rank the document sentences to get effective summaries. After all, sentence ranking is always the issue of most concern in summarization tasks.

Some existing CODS methods firstly derived the main information from comments with different strategies, such as, by identifying different relations between the comments to choose the most popular ones [3,4]. After this, based on the selected comments, chose top-N document sentences with different ranking strategies and generated the summaries. However, these methods ignored a key point that comments are supplement of document content but cannot always cover all aspects. Indeed, a large proportion of the comments are not directed to the content of the document or just act on a few aspects of the document. The selected comments may be related or unrelated to the main aspect of the document. So it is not always effective to generate the summaries based on these comments. Moreover, these works didn't consider that different comments might have different effects on the final ranking results.

Inspired by these, we propose a novel method called MultiAspectCoRank for CODS in this study. Firstly, in order to obtain useful information from comments and weaken the negative effect of the unrelated or one-sided ones, we try to fuse the document sentences and its comments together to get multiple aspects. Each aspect can be regarded as a part of the document. With these aspects, we can get multiple ranking lists, indicating different kinds of candidate summaries based on different angles. To overall consider the influence of different aspects on the ranking results, we use the co-feedback ranking framework, selecting top-N sentences from each ranking list and providing them as feedback to update other aspect based ranking lists in order to boost the ranking performance. This process continues iteratively until the top-N of each ranking list are unchanged. We get the final result by integrating these different ranking lists according to the weights of different aspects. Compared with previous work, our proposed MultiAspectCoRank method performs better on a real-world blog dataset with manually labeled sentences.

The rest of this paper is organized as follows. We briefly review the related work in Section 2. In Section 3 we present the details of multi-aspect co-feedback

ranking method for CODS. Experiments and results are discussed in Section 4. Finally we draw a conclusion of this study in Section 5.

2 Related Work

Comments-oriented document summarization is developed from automatic document summarization, so the related work will be introduced in two parts. Firstly we describe some representative summarization methods and then the special work about comments-oriented document summarization is presented briefly.

2.1 Automatic Document Summarization (ADS)

Traditional document summarization is a process to generate a summary by reducing documents in size while retaining the main characteristics of the original documents. In order to achieve this goal, different features and ranking strategies have been studied.

Feature-based sentence-ranking approaches are widely used in automatic document summarization. Radev et al. [10] proposed a centroid-based method, which implemented MEAD as a centroid-based summarizer by combining several predefined features including TF*IDF, cluster centroid and position to score the sentences. Lin and Hovy [5] built the NeATS multi-document summarization system using term frequency, sentence position and stigma words. Nenkova et al. [8] proved that high-frequency words were significant in reflecting the focus of documents. Ouyang et al. studied the influence of different word positions in summarization [9].

Graph-based ranking algorithms nowadays are successfully applied in summarization. LexPageRank [1] is the representative work which is based on the PageRank algorithm. Graph-based ranking algorithms take global information into consideration rather than rely only on vertex-specific information, therefore have been proved successful in document summarization. Some methods have been proposed to extend the conventional graph-based models recently including multi-layer graph incorporated with different relationship [12], multi-modality graph based on the manifold-ranking method [13] and DivRank [7] introducing the time-variant matrix into a reinforced random walk to balance prestige and diversity.

2.2 Comments-Oriented Document Summarization (CODS)

Comments-oriented document summarization (CODS) is a special task developed from traditional document summarization. The task is to summarize a blog post using the information hidden in its comments. Yang et al. [15] explored an approach by modeling web documents and social contexts into a unified framework. They proposed a dual wing factor graph (DWFG) model, utilizing the mutual reinforcement between web documents and their associated social contexts to generate summaries. Hu et al. [4] firstly scored the importance of each

comment based on three relations (namely, topic, quotation, and mention) with a graph based method and a tensor based method, and then extracted sentences from the given web document with two approaches: feature-biased approach and uniform-document approach. For the former one, words appearing in comments but not in the document do not contribute to scoring sentences, which is more tolerant to noise in comments. For the latter one, when there is no or very few comments, the problem naturally degrades to single document summarization.

3 Approach Details

In this section, we propose a multi-aspect co-feedback ranking model (MultiAspectCoRank), which utilizes the mutual reinforcement on the correlations based on each aspect. This MultiAspectCoRank model incorporates both the document content and its corresponding comments together to generate a high quality summary.

3.1 Multiple Aspects Extraction

Comments represent the readers' feedback about the web document. However, not all comments are directed to the content of the document. In order to derive useful information from comments, we try to fuse the document sentences and its affiliated comments together and extract multiple aspects with clustering technique. In this way, most related comments will be clustered with document sentences together.

Affinity propagation is a graph based cluster algorithm [2]. Compared with traditional clustering algorithms such as K-means, singular value decomposition (SVD), graph-based approach using affinity propagation performs best in clustering short text data with minimal cluster error [11]. Affinity propagation takes as input a collection of real-valued similarities between data points. In order to identify the exemplars (centers of clusters) of a document, two kinds of message exchanged between data points: one is called "responsibility" $\gamma(i, j)$, sent from data point i to candidate exemplar point j , the other is "availability" $\alpha(i, j)$, sent from candidate exemplar point j to point i . To begin with, the availabilities are initialized to zero: $\alpha(i, j) = 0$. Then the responsibilities are computed using the rule:

$$\gamma(i, j) \leftarrow sim(i, j) - \max_{j' \neq j} \{\alpha(i, j') + sim(i, j')\} \quad (1)$$

where $sim(i, j)$ represents the similarity between data point i and j . Whereas the above responsibility update lets all candidate exemplars compete for ownership of a data point, the following availability update gathers evidence from data points as to whether each candidate exemplar would make a good exemplar:

$$\alpha(i, j) \leftarrow \min\{0, \gamma(j, j) + \sum_{i' \notin \{i, j\}} \max\{0, \gamma(i', j)\}\} \quad (2)$$

Given a web document D , we denote the graph of the whole sentences set as $G = (S, E_{ss})$ to represent the whole document including comments. $S = \{s_i | 0 \leq i \leq n\}$ is the set of vertices in the graph and stands for the set of both the document sentences and its affiliated comment sentences, and $E_{ss} = \{e_{ij} | s_i, s_j \in S, i \neq j\}$ corresponds to the relationship between each pair of sentences. Each e_{ij} is associated with a weight ω_{ij} which indicates the similarity of the pair of sentences. The weight is computed by using the standard cosine measure between two sentences as follows:

$$\omega(i, j) = sim_{cosine}(s_i, s_j) = \frac{\vec{v}_i \cdot \vec{v}_j}{|\vec{v}_i| \times |\vec{v}_j|} \quad (3)$$

where \vec{v}_i and \vec{v}_j are the term vectors corresponding to the sentence s_i and s_j , respectively. The graph we propose to build is undirected so we have $\omega_{ij} = \omega_{ji}$ here and define $\omega_{ii} = 0$ to avoid self transition. $TF-ISF$ (term frequency - inverted sentence frequency) value of each term is applied to describe the elements in the sentence vector. Then the weight of each pair of sentences in the web document can be denoted as a symmetric matrix W , which is used as the input of the affinity propagation. For the affinity propagation with W and a specified number of exemplars K , the message passing procedure will be terminated after a fixed number of iterations. And then we get the result of K clusters. Each cluster is a sentence list standing for an aspect of the document.

3.2 Sentence Ranking Based on Each Aspect

As we get K aspects, each aspect can be viewed as a part of the document. We need to estimate the probability score $p(s|A)$ of a sentence s being selected as a summary sentence given an aspect A . Instead of simply comparing the relevance between the sentence s and the aspect A , we take this as a task based on the whole background, where both the similarities between the sentence s and other sentences in the document and the similarity between s and the aspect A are considered. Formally, $p(s|A)$ is computed by the following formula:

$$p(s|A) = d \sum_{z \in S_D} \frac{sim(s, z)}{\sum_{q \in S_D} sim(q, z)} p(z|A) + (1 - d) \frac{rel(s, A)}{\sum_{z \in S_D} rel(z, A)} \quad (4)$$

where A is a set of sentences standing for an aspect, and S_D is the set of all sentences in the document, d is a trade-off parameter in the interval $[0, 1]$, which is used to specify the relative contribution of the two parts in Eq.(4). For bigger value of d , more importance is given to the similarities between the sentence s and other sentences in the document D than that between s and the aspect A . The denominators in both parts are used for normalization. The formula in Eq.(4) can be written as:

$$p(k+1) = M^T p(k) \quad (5)$$

$$M = dC + (1 - d)R \quad (6)$$

where M , C and R are all square matrices. Elements in C represent the similarities between sentences in the document. Elements in R represent the relevance

between each sentence in the document and each aspect. k represents the k th iteration, and $p = [p_1, p_2, \dots, p_N]^T$ is the vector of sentence ranking scores that we are looking for, which corresponds to the stationary distribution of the matrix M . The iteration is guaranteed to converge to a unique stationary distribution given that M is a stochastic matrix.

To calculate the similarities between the sentences in the document, we use the Cosine similarity. To calculate the relevance between s and an aspect A , we use the average Cosine similarity as follows:

$$rel(s, A) = \frac{\sum_{a \in A} sim_{cosine}(s, a)}{count(A)} \quad (7)$$

where a represents a sentence belongs to the aspect A , $count(A)$ stands for the number of sentences in aspect A .

In order to initialize each $p(s|A)$, we can regard each aspect as a short document and construct a language model to estimate it using Dirichlet prior smoothing as follows:

$$p(s|A) = \sum_{w \in s} p(w|A) \quad (8)$$

$$p(w|A) = \frac{c(w, A) + u_s * p(w|B)}{|A| + u_s} \quad (9)$$

where $|A|$ is the length of the sentence aspect A , $c(w, A)$ is the count of word w in A , u_s is the smoothing parameter, $p(w|B)$ is the background model used as smoothing factor. Generally $p(w|B)$ is estimated by the whole document D , using following formula:

$$p(w|B) = \frac{c(w, D)}{|D|} \quad (10)$$

After estimating the score of each sentence in the document, we can get K lists of scores $list_i = \{r_i(s_j) | 1 \leq j \leq |S_D|, 1 \leq i \leq K\}$, where $r_i(s_j)$ represents the ranking result of sentence s_j in ranking list $list_i$, $|S_D|$ is the number of sentences in the document, no comments included, K is the number of aspects.

3.3 Co-feedback Ranking Based on Multiple Aspects

Each ranking list indicates a kind of candidate summaries from one angle, but cannot determine the final ranking results. In this subsection, we are aiming to rank all the document sentences based on the K groups of ranking lists and selecting the top- N ranked ones to generate summaries. Each aspect can be regarded as a part of the document. The summary we are aiming to generate should base on the whole document. So how to combine the ranking scores based on each aspect to get an overall score as the ranking criterion is the key problem to solve. Traditional strategy is to combine different ranking lists into a unified one with a linear function. However it fails to make full use of the information provided by the different ranking lists and neglects the interactions among them

before combination. Also we believe that each aspect based ranking list is able to provide valuable information to help promote other aspects based ranking performance with the mutual ranking refinement, which, in turn, may lead to an overall improvement in ranking.

Co-feedback ranking framework [14] is a good mutual reinforcement method to combine multiple ranking results into a unified one. In this paper, we select the top- N ranked sentences from each ranking list as feedback to update other aspect based ranking results according to the relevance of sentences. So we update the ranking result of s_j in ranking list $list_i$ as follows:

$$r_i^{t+1}(s_j) = \eta r_i^t(s_j) + (1 - \eta) \pi_{ij}^t, \quad 1 \leq i \leq K, 1 \leq j \leq |S_D| \quad (11)$$

where η is a balance parameter which can be viewed as the proportion of the dependence of the new ranking results on its initial ranking results. π_{ij} is the feedback value from other ranking lists to s_j of ranking list $list_i$. t represents the t th iteration. π_{ij}^t is computed as follows:

$$\pi_{ij}^t = \sum_{k=1, k \neq i}^K \frac{\alpha_k}{\sum_{k=1, k \neq j}^K \alpha_k} \pi_{ikj}^t \quad (12)$$

$$\pi_{ikj}^t = \sum_{l=1}^N \frac{\sum_{h=0}^N h - l}{\sum_{h=0}^N h} sim(s_j, \tau_k^N) \quad (13)$$

where π_{ikj}^t represents the feedback value on sentence s_j of ranking list $list_i$ from ranking list $list_k$. α_k represents the number of sentences in aspect k . τ_k^N represents the top N sentences selected from ranking list $list_k$. $sim(s_j, \tau_k^N)$ is defined as the similarity between s_j and the top N sentences from ranking list $list_k$. In this paper N is set to 10 as 10 sentences are basically sufficient for the summarization task we work on. This process continues iteratively when the top N results of each ranking list don't change. We get the final ranking results based on these ranking lists as follows:

$$r(s_j) = \sum_{i=1}^K \frac{\alpha_i}{|S_D|} r_i(s_j) \quad 1 \leq j \leq |S_D| \quad (14)$$

This co-ranking method is designed based on the principle that the new ranking result of the sentence s from one aspect consists of two parts: one is the initial ranking result based on the aspect, the other part is the similarities between s and the top N feedback sentences provided by other ranking lists. The top N ranked sentences based on one aspect are presumed to be highly supported by that aspect, which are more likely to be chosen as summary sentences. So these top N sentences are selected as feedback to promote the weight of these sentences in other ranking lists. Through this co-feedback and mutual reinforcement, these ranking lists are considered to be more reliable ranking results.

After we get the final ranking lists, we apply the variant version of MMR algorithm proposed in [12]. This method is to penalize the sentences that highly overlap with the sentences that have been chosen as the summary. In this way, we can choose more informative but less redundant sentences as the final summary.

4 Experiments

4.1 Dataset

To evaluate our method for comments-oriented document summarization, we conduct our experiments on the dataset used in [4]. This dataset was collected from two famous blogs, i.e., Cosmic Variance¹ and IEBlog², both receiving large number of comments. To guarantee the validity of the data, we randomly picked up 50 posts from each blog, and constructed our experiment dataset with these 100 articles. Each article consists of two parts, document content and its affiliated comments. Table 1 gives the brief description of the 100 posts. To generate reference summaries, each post has 7 labeled sentences by 4 human summarizers.

Table 1. A brief description of the dataset

Source	Cosimic Variance and IEBlog
Number of blog posts	100
Average sentence number per post	22.22
Average sentence length	19.71
Average comment number per post	26.04

4.2 Performance Metric

We use the ROUGE³ [6] (Recall Oriented Understudy for Gisting Evaluation) toolkit to evaluate our proposed method, which has been widely applied for summarization evaluation. It evaluates the quality of a summary by counting the overlapping units between the candidate summary and reference summaries. There are many kinds of ROUGE metrics to measure the system-generated summarization such as ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-U, of which the most important one is ROUGE-N with 3 sub-metrics: precision, recall, and F-score.

$$ROUGE - N = \frac{\sum_{S \in RS} \sum_{N-gram \in S} Count_{match}(N-gram)}{\sum_{S \in RS} \sum_{N-gram \in S} Count(N-gram)} \quad (15)$$

where RS represents the reference summaries. $N\text{-gram} \in RS$ in the metrics denotes the N-grams in reference summary. $Count_{match}(N\text{-gram})$ is the maximum

¹ <http://cosmicvariance.com>

² <http://blogs.msdn.com/ie/>

³ <http://www.isi.edu/licensed-sw/see/rouge/>

number of N-grams co-occurring in the candidate summary and in the set of reference summaries. $Count(N\text{-}gram)$ is the number of N-grams in the reference summaries.

The ROUGE toolkit can report separate scores for 1, 2, 3, and 4-gram. In the experimental results we report four ROUGE F-measure scores: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based), ROUGE-4 (extension of ROUGE-S, which is the skip-bigram co-occurrences statistics), and ROUGE-W (based on the weighted longest common subsequence) metrics. The higher the ROUGE scores, the similar the two summaries are.

Table 2. Results of different systems

Methods	ROUGE-1	ROUGE-2	ROUGE-4	ROUGE-W
Random	0.3944	0.2984	0.2705	0.3075
LexPageRank	0.5961	0.4593	0.4627	0.5601
LexPageRank**	0.5731	0.4386	0.4416	0.5174
MultiAspectCoRank	0.6295	0.4911	0.4893	0.5847

4.3 Experimental Results

Given a web document and its corresponding comments, we first decompose them into sentences and then remove the stop-words and words stemming are performed. After these steps, we implement the following algorithms which are widely used in comments-oriented document summarization. We conduct the same preprocessing for all algorithms for fairness.

- Random (system1): The method selects sentences randomly for the sentence set including comments.
- LexPageRank [1] (system2): This system applies Cosine similarity in standard LexPageRank method based on the document content sentence set (no comments is given) to summarize the document.
- LexPageRank** (system3): This system applies Cosine similarity in standard LexPageRank method based on the whole document sentence set (including comments) to generate the summary.
- MultiAspectCoRank (system4): As we have showed, this method extracts multiple aspects with affinity propagation firstly, and then utilizes mutual reinforcement co-feedback ranking framework based on these aspects to promote the ranking performance.

Table 2 and Fig 1 show the performance of these systems on a same dataset. The parameters of MultiAspectCoRank approach are set as follows: the number of aspects $K = 5$, the balance parameter $u_s = 0.1$, $d = 0.6$ and $\eta = 0.4$. From Table 2 and Fig 1 we have following observations:

- Generally, the Random gets the worst performance;
- From the results of system 2 and system 3, both of these two systems use LexPageRank, but the results of system 2 in which no comments is given is better than those of system 3 which combines document contents and comments

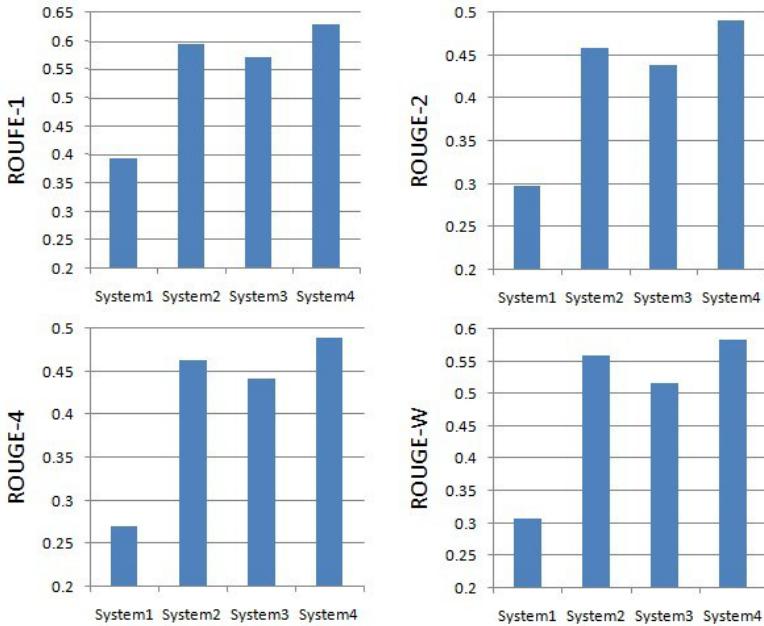


Fig. 1. Comparison results in ROUGE

together. This is mainly because most of the comments are not directed to the content of the post and comments are not equally useful for identifying important document sentences.

- From Fig 1, we can see that on the same dataset, our system (system 4) outperforms other systems on different evaluation measures, which indicates the effectiveness of the proposed MultiAspectCoRank model.

All the comparison results suggest that it's effective to derive useful information from comments by fusing the document sentences and its corresponding comments together and detecting multiple aspects from them. And it's better to promote the performance when utilizing mutual reinforcement co-feedback ranking method on the same data based on multiple aspects.

4.4 Parameter Tuning

There are mainly four parameters in our proposed method: number of clusters K , balance parameter u_s , d and η . In this section, we are trying to evaluate the influence of these four parameters on the experimental results respectively. First we compare the four kinds of ROUGE measure results with aspect number K ranging from 3 to 11 in Fig 2. It is observed that when the aspect number K is 7 the performance is best. In the same way, the four kinds of ROUGE measure results with balance parameter u_s , d and η are shown in Fig 2. It is obvious that the value of parameter u_s has less effect on the performance of our proposed

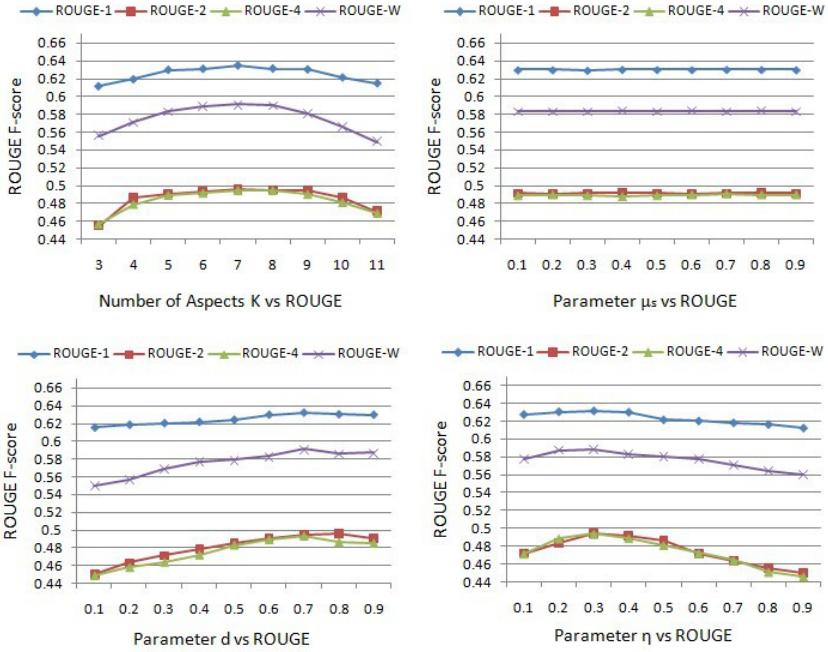


Fig. 2. Parameters vs ROUGE

method, and when the balance parameter d is set to 0.7 and η is set to 0.3 this method performs best.

5 Conclusion

In this work, we propose a novel model named MultiAspectCoRank for comments-oriented document summarization task. More specifically, by fusing the document content and its comments together, we extract multiple aspects from the given web document, and then utilize mutual reinforcement co-feedback ranking method to promote the performance. Experimental results on a set of real-world blog data prove the effectiveness of our proposed method in improving the summary quality. In future work, we will try to find more and deeper relationships between document contents and comments.

Acknowledgments. We thank the anonymous reviewers for their valuable and constructive comments. This work is financially supported by NSFC Grant 61272340.

References

1. Erkan, G., Radev, D.: LexpageRank: Prestige in multi-document text summarization. In: Proceedings of EMNLP, vol. 4 (2004)
2. Frey, B., Dueck, D.: Clustering by passing messages between data points. *Science* 315(5814), 972–976 (2007)
3. Hu, M., Sun, A., Lim, E.: Comments-oriented blog summarization by sentence extraction. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, pp. 901–904. ACM (2007)
4. Hu, M., Sun, A., Lim, E.: Comments-oriented document summarization: understanding documents with readers feedback. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 291–298. Citeseer (2008)
5. Lin, C., Hovy, E.: From single to multi-document summarization: A prototype system and its evaluation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 457–464. Association for Computational Linguistics (2002)
6. Lin, C., Hovy, E.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, vol. 1, pp. 71–78. Association for Computational Linguistics (2003)
7. Mei, Q., Guo, J., Radev, D.: Divrank: the interplay of prestige and diversity in information networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1009–1018. ACM (2010)
8. Nenkova, A., Vanderwende, L., McKeown, K.: A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 573–580. ACM (2006)
9. Ouyang, Y., Li, W., Lu, Q., Zhang, R.: A study on position information in document summarization. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pp. 919–927. Association for Computational Linguistics (2010)
10. Radev, D., Jing, H., Styś, M., Tam, D.: Centroid-based summarization of multiple documents. *Information Processing & Management* 40(6), 919–938 (2004)
11. Rangrej, A., Kulkarni, S., Tendulkar, A.: Comparative study of clustering techniques for short text documents. In: Proceedings of the 20th International Conference Companion on World Wide Web, pp. 111–112. ACM (2011)
12. Wan, X.: Document-based hits model for multi-document summarization. In: Ho, T.-B., Zhou, Z.-H. (eds.) PRICAI 2008. LNCS (LNAI), vol. 5351, pp. 454–465. Springer, Heidelberg (2008)
13. Wan, X., Yang, J., Xiao, J.: Manifold-ranking based topic-focused multi-document summarization. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 2903–2908. Morgan Kaufmann Publishers Inc. (2007)
14. Wei, F., Li, W., He, Y.: Co-feedback ranking for query-focused summarization. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pp. 117–120. Association for Computational Linguistics (2009)
15. Yang, Z., Cai, K., Tang, J., Zhang, L., Su, Z., Li, J.: Social context summarization. In: Proceedings of the 34th ACM SIGIR Conference (2011)

Image Annotation with Weak Labels

Feng Tian^{1,2,*} and Xukun Shen¹

¹ State Key Laboratory of Virtual Reality Technology and Systems,
Beihang University, 100191 Beijing, China

² School of Computer and Information Technology,
Northeast Petroleum University, 163318 Daqing, China
{tianfeng,xkshen}@vrlab.buaa.edu.cn

Abstract. In this paper, we address the problem of image annotation when the given labels of training image are incomplete, inaccurate, and unevenly distributed, in the form of weak labels, which is frequently encountered when dealing with large scale web image training set. We introduce a progressive semantic neighborhood learning approach that explicitly addresses the challenge of learning from weakly labeled image by searching image's semantic consistent neighborhood. Neighbors in image's semantic consistent neighborhood have global similarity, partial correlation, conceptual similarity along with semantic balance. We also present an efficient label inference algorithm to handle noise by minimizing the neighborhood reconstruction error. Experiments with different data sets show that the proposed framework is more effective than the state-of-the-art algorithms in dealing with weakly labeled datasets.

Keywords: label set relevance, web image annotation, image label.

1 Introduction

Traditional image annotation studies, a basic assumption is that all the proper labels of every training image are given and correct. In real environment, this assumption hardly holds since getting all the proper labels is usually expensive, time consuming and people usually add a few labels, rather than an exhaustive list of relevant terms. Moreover, not all of the labels are relevant to the image content ,for example, images labeled with "car" might be taken from a car, rather than depicting one. It is evident that this scenario is quite different from the classic image annotation setting where all proper labels for training data are assumed to be given. Images in benchmark set are also usually weakly labeled(showed in Table 1). Meanwhile, large variations in the frequency of different labels can reduce the performance of the labeling method on the low-frequency labels. E.g., in an experiment on the Corel5K dataset, we found that for the 20% least frequent labels, JEC [1] achieves an F-score of 19.7%, whereas it gives reasonably good performance for the 20% most frequent labels with F-score being 50.6%. In this work, the meaning of the terminology "weak

* Corresponding author. This work is supported by Scientific Research Fund of Heilongjiang Provincial Education Department(NO:12511011,12521055).

Table 1. Weak label image (the missing labels are highlighted by bold font, the content unrelated labels are italicized)

				
field horses mare fence mountain <i>travel vacation</i> bear river czech bridge foals range sky Nile sky reflection water <i>charles lights</i> tree airplane sailboat sea black night				

labels” is threefold: (1) the given labels may be incomplete, namely only a subset of labels are attached to images according to the ground truth; (2) even for the labels provided, there may be noisy labels; (3) there is large variations in the frequency of different labels (semantic imbalance). Image annotation from weakly labeled dataset is important since weakly-labeled problems are prevalent in the popular datasets as well as real-world environment. In [2], the authors showed performance improvement where for each training instance, only one of its class assignments is correct. In [3], a hybrid model framework for utilizing partially labeled data that integrates a generative topic model for image appearance with discriminative label prediction is explored. In [4], the author focuses on removing false class assignments for training set. In [5], the author proposed ranking based multi-label learning to learn from incompletely data. Our work is more comprehensive and address a more realistic and challenging scenario where the datasets seriously suffer from weakly labeled issues.

2 Our Approach

Based on the idea that negative impact of the weak label can be reduced under the guidance of neighbors, the training image’s labels are replenished by minimizing the label’s weighted error function, then “semantic balanced neighborhood” is set up based on the replenishing labels to address the large differences in these label’s frequency. Linear metric embedded in multiple label information is learned to obtain the consistency of distance measure and image semantic. Then the images’ partial correlation is obtained by image’s nonnegative sparse linear combination between neighbors. The neighbors in the final neighborhood have higher global similarity, partial correlation and conceptual similarity along with semantic balance. Label prediction is performed in the neighborhood by minimizing label’s reconstruction error loss, and noise labels are handled by two regulation terms.

2.1 Semantic Balanced Neighborhood

By semantic balanced neighborhood (short for BN), we mean, for a given image, there should not have large differences in the frequency of different labels in

it's neighborhood. Considering labels for training image may be incomplete, we replenish missing labels firstly. Denote vocabulary as $C = \{c_1, c_2, \dots, c_q\}$ and training set $L = \{(x_1, y_1), \dots, (x_l, y_l)\}$, m represents the dimensionality of features, $y_i = (y_{i1}, \dots, y_{iq}) \in \{0, 1\}^q$ is the corresponding label vector, $y_{ij} = 1$ if the i -th image has the j -th label and $y_{ij} = 0$ otherwise. $Y = [y_1, \dots, y_l]^T$ be the corresponding label indicator matrix. We want to learn a replenished function $f : L \rightarrow R^q$ where $f_i = [f_{i1}, f_{i2}, \dots, f_{iq}]^T$, f_{ij} denotes the value of function output of i -th image, and we use matrix $F = [f_1, f_2, \dots, f_l]$ to present the replenished label matrix. The error function is $E(f) = E_1(f) + \lambda E_2(f)$, where $\lambda \geq 0$ is a controlling parameter. Thus, the optimization problem is:

$$\min_f \left\{ \frac{1}{2} \sum_{j=1}^q \sum_{i=1}^l u_{ij} (y_{ij} - f_{ij})^2 + \frac{1}{2} \lambda \sum_{i=1}^l \sum_{j=1}^l w_{ij} \left\| \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right\|^2 \right\}$$

$E_1(f)$ represents the weighted error function, u_{ij} represents the weight between sample x_i and j -th label, $u_{ij} = 1$ if $y_{ij} = 1$, and τ otherwise ($0 \leq \tau \leq 1$). Minimizing $E_1(f)$ is equivalent to requiring the output of f is similar to the original labels and can replenish the missing labels. Minimizing the second term is equivalent to requiring the smoothness output of f on each sample's neighbor according to their similarity. The approximate optimal solution can be derived by least squares. Based on the replenished labels, image's BN is constructed as follows. Let $L_i \subseteq L (\forall i \in \{1, 2, \dots, q\})$ be the subset of training data that contains all the images annotated with the label c_i , we consider it as a semantic group. Given an image x , from each semantic group we pick k_2 images that are most similar to x and form corresponding sets $L_{x,i} \subseteq L_i$. Thus, each $L_{x,i}$ contains images that are most informative in predicting the probability of the label c_i for x . We merge them all to form the semantic balanced neighborhood as $BN(x) = \{L_{x,1} \cup \dots \cup L_{x,q}\}$. It can be easily noted that in $BN(x)$, each label appears (at least) k_2 times, thus addressing the semantic imbalance issue.

2.2 Semantic Consistent Neighborhood

By semantic consistent neighborhood(short for CN), we mean, the neighbors should have both the global similarity and partial correlation along with conceptual similarity. We select the partial correlated neighbors in target image's BN by sparse representation. Note that, from signal reconstruction point of view, when target signal is reconstructed from signals in different subspace(semantic subspace), the reconstruction coefficients have lost their physical meaning. It is obvious that we cannot guarantee that sample's in BN are all semantically similar, since image pair's semantic similarity depends on the corresponding label set instead of single label. As shown in figure1(a), x_p 's semantically similar neighbors (denoted by circle) and neighbors that are semantically dissimilar neighbors(denoted by square) are all in x_p 's BN. If semantically dissimilar neighbors lie outside smaller radius with a margin of at least one unit distance, as shown in figure2(b), then we can reconstruct x_p by neighborhood (b). Let a and

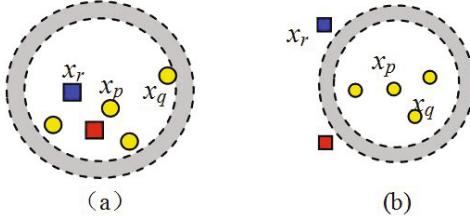


Fig. 1. Schematic illustration of one input's neighborhood

a and b be two training images, and each represented by n features $\{f_A^1, \dots, f_A^n\}$ and $\{f_B^1, \dots, f_B^n\}$, where $\sum m_i = m$.

$$\tilde{d}(a, b) = \sum_{i=1}^n w(i) \sum_{j=1}^{m_i} u^i(j) \cdot dist_{AB}^i(j) \quad (1)$$

u^i and w are usually taken as a non-negative normalized unit vector, u^i can be assigned appropriate weights to individual dimensions of a feature vector in the feature space, w is to optimally combine multiple feature distances. Given an image x_p , along with its label vector y_p , we want to learn weights such that its target neighbor x_q from the semantic groups $\{L_{x_p, r}\}_r$ are pulled closer, x_r from the remaining semantic groups are pushed far. That is, minimize the error function:

$$\operatorname{argmin}_{w, u} \sum_{pq} \eta_{pq} \lambda_{pq} \tilde{d}(x_p, x_q) + \mu \sum_{pqr} \eta_{pq} (1 - \lambda_{pr}) [1 + \tilde{d}(x_p, x_q) - \tilde{d}(x_p, x_r)]_+$$

Here, μ is the controlling parameters, $[z]_+ = \max(0, z)$ is the hinge loss, λ_{pq} and λ_{pr} scale the error loss depending on the overlap between the label sets of images. We solve it by alternatively using stochastic sub-gradient descent and projection steps (similar to Pegasos [6]) to obtain an approximate optimal solution of w and u^i . Then, given image x_i , we find it's k nearest neighbor by Equation (1) to construct x_i 's local overcomplete dictionary, where $i_p \in \{1, \dots, l\}$, $p \in \{1, \dots, k\}$. i_p is the reconstruction coefficients vector for x_i . Note that negative coefficient has not explicit meaning to describe semantic, so we reformulate the reconstruction relationship as $x_i = B_i \alpha_i + \zeta$, where $\alpha_i(p) \geq 0$ and $\sum_{p=1}^k \alpha_i(p) = 1$. Let non-negative term ζ^+ , noise term $\zeta = \zeta^+ - \zeta^-$, $|\zeta| = \zeta^+ + \zeta^-$. Then we can solve $\min_{\alpha_i} \lambda \|\alpha_i\|_1 + \frac{1}{2} \|x_i - B_i \alpha_i\|_2^2$ s.t. $\alpha_i \geq 0$ where $x_i = [x_i \ 1]^T$, $\alpha_i = [\alpha_i \ \zeta^+ \ \zeta^-]^T$. $B_i = \begin{bmatrix} B_i & I_m & -I_m \\ E_{1 \times k} & 0_{1 \times m} & 0_{1 \times m} \end{bmatrix}$, the controlling term $\lambda = 2\|B_i x_i\|_\infty$, the problem can be solved efficiently using L1 optimization toolbox like YALL. Then, x_i is represented by a sparse linear combination of it's neighbors, and it's semantic consistent neighborhood(CN) is composed by the neighbors x_{i_p} where $\alpha_i(p) > 0$. Let $C = [c_{ij}]$ denotes neighborhood weight matrix, where $c_{ij} = \alpha_i(p)$.

2.3 Label Inference in CN

Let $f = [f_L \ f_U]^T$ be label matrix of all samples, where f_L represents the training set's label matrix, f_U represents the unlabeled ones' label matrix(initialized by zero). Assuming that each image's label vector can be reconstructed by it's neighbors in it's CN, while the reconstruction coefficients are the same as their visual reconstruction coefficient. Thus we can predict the labels of the unlabeled samples by the weight in neighborhood matrix C . This prediction is based on the assumption that the weight c_{ij} reflects the likelihood for sample x_i to have the same label as sample x_j . So the labels of the unlabeled samples can be inferred by minimizing label reconstruction error as follows:

$$E(f) = \sum_{i=1}^n \|f_i - \sum_{j \neq i} c_{ij} f_j\|^2, s.t. f_i = y_i \quad (2)$$

where y_i is the replenished label vector of x_i . We use generalized minimum residual method (GMRES [7]) to obtain an approximated solution. As aforementioned, the associated labels are often incomplete and imprecise, so the training labels cannot be fixed during the inference process as in Equation (2) should be refined. However, the training labels should be consistent with the original labels to a certain extent. So the optimization target should be

$$\min_f \left\{ \|f - Cf\|^2 + \lambda_1 \|f_L - \hat{f}_L\|^2 + \lambda_2 \|\hat{f}_L - Y\|_1 \right\}$$

where f_L is the training images labels that are propagated, and \hat{f}_L denotes the ideal label vector of the training images. The first term of this formula is the same as in Equation (2). The second term enforces the ideal labels of the training images to be consistent with the labels propagated. The third term constrains that only a limited number of labels are noisy or imprecise.

3 Experiments

We validate the effectiveness of our proposed approach on IAPR-TC12,ESPGAME and FLICKR2.5M datasets. Each image is annotated with

Table 2. Experimental results on four dataset

Method	IAPR				ESP				FLI			
	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+
SML[8]	0.21	0.23	0.220	201	0.16	0.17	0.165	195	0.15	0.16	0.155	278
JEC [1]	0.28	0.29	0.285	250	0.22	0.25	0.234	224	0.20	0.21	0.205	355
Tagprop(ml) [9]	0.48	0.25	0.329	227	0.49	0.20	0.284	213	0.43	0.17	0.244	363
Tagprop(ml+s)[9]	0.46	0.35	0.398	266	0.39	0.27	0.319	239	0.34	0.25	0.288	403
GS [10]	0.32	0.29	0.304	252	0.36	0.24	0.288	226	0.29	0.22	0.250	375
SNLWL	0.52	0.37	0.432	276	0.51	0.30	0.378	249	0.48	0.29	0.362	427

5 most relevant keywords. Results are summarized in Table 2. From the results we find that our method significantly improve over the current state-of-the-art. On ESP-GAME image dataset, we achieve 30.8% and 11.1% performance improvement in terms of precision and recall. On FLICKR, we achieve 41.2% and 24% performance improvement in terms of precision and recall.

4 Conclusion

Image annotation with weakly labeled images is important since weakly-labeled problems are prevalent in the popular web datasets as well as real-world environment. Experiments show that the proposed framework is more effective than state-of-the-art over web image dataset.

References

1. Makadia, A., Pavlovic, V., Kumar, S.: A new baseline for image annotation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part III. LNCS*, vol. 5304, pp. 316–329. Springer, Heidelberg (2008)
2. Nguyen, N.: Classification with partial labels. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 551–559. ACM Press, New York (2008)
3. He, X.: Learning hybrid models for image annotation with partially labeled data. In: *Conference on Neural Information Processing Systems*, pp. 625–632 (2008)
4. Fan, J.: Harvesting large-scale weakly-tagged image databases from the web. In: *International Conference on Computer Vision*, pp. 802–809. IEEE Computer Society Press, Los Alamitos (2010)
5. Bucak, S.: Multi-label learning with incomplete class assignments. In: *International Conference on Computer Vision*, pp. 2801–2808. IEEE Computer Society Press, Los Alamitos (2011)
6. Shwartz, S.: Pegasos: Primal estimated sub-gradient solver for svm. In: *International Conference on Machine Learning*, pp. 807–814 (2007)
7. Saad, Y.: Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. In: *International Conference on Computer Vision; SIAM Journal on Scientific and Statistical Computing* 7, 856–869 (1986)
8. Carneiro, G.: Supervised learning of semantic classes for image tagging and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 394–410 (2007)
9. Guillaumin, M.: Tagprop: Discriminative Metric Learning in Nearest Neighbor Models for Image Auto-tagging. In: *International Conference on Computer Vision*, pp. 309–316. IEEE Computer Society Press, Los Alamitos (2009)
10. Zhang, S.: Automatic image annotation using group sparsity. In: *International Conference on Computer Vision*, pp. 3312–3319. IEEE Computer Society Press, Los Alamitos (2010)

Scan and Join Optimization by Exploiting Internal Parallelism of Flash-Based Solid State Drives

Wenyu Lai, Yulei Fan, and Xiaofeng Meng

Renmin University of China, Beijing, China

{xiaolai913, fy1815, xfmeng}@ruc.edu.cn

Abstract. Nowadays, flash-based solid state drives (SSDs) are gradually replacing hard disk drives (HDDs) as the primary non-volatile storage in both desktop and enterprise applications because of their potential to speed up performance and reduce power consumption. However, database query processing engines are designed based on the fundamental characteristics of HDDs, so they may not benefit immediately from SSDs. Previous researches on optimizing database query processing on SSDs have mainly focused on leveraging the high random data access performance of SSDs and avoiding slow random writes whenever possible. However, they fail to exploit the rich internal parallelism of SSDs. In this paper, we focus on exploiting rich internal parallelism of SSDs to optimize scan and join operators. Firstly, we detect internal parallelism of SSDs seemed as black boxes. Then we propose a parallel table scan operator called ParaScan to take full advantage of internal parallelism of SSDs. Based on ParaScan, we also present an efficient parallel join operator called ParaHashJoin to accelerate database query processing. Experimental results on TPC-H datasets show that our ParaScan on SSD significantly outperforms the traditional table scan on SSD by 1X, and ParaHashJoin is 1.5X faster than traditional hash join operator especially when join selectivity is small.

Keywords: Flash-based SSDs, Internal Parallelism, Query Processing.

1 Introduction

Flash-based solid state drives (SSDs), as a new type of non-volatile storage, are gradually replacing the central role of traditional magnetic hard disk drives (HDDs). More and more portable devices are equipped with SSDs to get excellent performance such as MacBook Air, Play Station and so on. Compared to HDDs, SSDs provide faster access speed, lower power consumption, lighter weight, smaller size and better shock resistance. In addition to these advantages, SSDs also have rich internal parallelism which provides us a chance to improve I/O bandwidth by doing parallel processing on them [1, 2]. Due to their potential to speed up applications and reduce power consumption, SSDs are expected to gradually substitute HDDs as mass storage media in large data centers.

The development of flash-based SSDs is also attracting researchers' interests to redesign various aspects of DBMS internals for SSDs such as storage management,

query processing and so on. Traditional query processing algorithms are mainly designed according to the mechanical traits of the disk, so they may benefit less or even nothing when SSDs are used as a simple drop-in replacement for disk for data analysis workloads in database [3]. Thus it is necessary to redesign the query processing algorithms to take full advantages of SSDs.

Scan and join are two important operators in DBMS. Scan operators are basic physical operations in database system and they can mainly divide into two categories: table scan and index scan [4]. Table scan reads data block one by one sequentially, which is good suit for HDDs. Lots of query processing operators often need to cooperate with table scan to accomplish a query such as sort, aggregation and join. Join operators are also basic operations in database system, but they are more complex. It is well-known that join operations can be expensive and can play a critical role in determining the overall performance of the DBMS. There are three classic ad hoc join algorithms in traditional DBMS, namely: nested loops join, sort-merge join and hash join [4]. In this paper, we mainly explore the optimization of table scan and hash join.

Due to rich internal parallelism of SSDs, it is possible for us to process multiple I/O requests at the same time on SSDs, that may offer us excellent IOPS (Input/Output Operations Per Second). However, the outstanding random I/O performance of SSDs will remain only a potential performance specification, unless DBMSs take advantage of internal parallelism and fully utilize the high IOPS.

In this paper, we investigate query processing methods that are better suited for the characteristics of SSDs, especially SSDs' internal parallelism. In particular, we focus on speeding up scan and join operations over tables stored on SSDs. Towards this goal, we make the following contributions.

- We detect and examine internal architecture of different kinds of SSDs and then propose a novel parallel table scan operator called ParaScan to take full advantage of internal parallelism of SSDs.
- Based on ParaScan, we present ParaHashJoin, an efficient parallel hash join operator which makes full use of internal parallelism of SSDs to accelerate join processing in a query plan.
- Experiment evaluation results on TPC-H datasets demonstrate that the proposed ParaScan and ParaHashJoin operators reduce the execution time of scan and join effectively.

The rest of the paper is organized as follows. In Section 2, we give the related work and compare our study with them. Section 3 introduces internal parallel architecture of SSDs at first and then discusses how to utilize this internal parallelism and why we should detect SSD internals. After that, we propose a parallel table scan operator and a parallel join operator in Section 4 and Section 5 respectively. Section 6 describes experimental results on TPC-H datasets, and we conclude in Section 7.

2 Related Work

In order to achieve high bandwidth and better IOPS, most modern SSDs adopt multi-channel and multi-way architecture and flash memory controller can access flash chips in parallel [5]. Therefore, we need to understand the impact of this parallelism inside SSDs to improve data processing performance. Chen et al. [2] studied on how to uncover internal parallelism features of SSDs and revealed that exploiting internal parallelism can significantly improve I/O performance. In the study of [6], researchers focused on finding an efficient way to generate parallel I/Os to access SSDs. By assessing different methods to create parallel I/O, authors suggest a new I/O request method and design a new B+-tree variant called PIO B-tree to exploit internal parallelism of SSDs. Other studies [7, 8] tried to improve SSD internal architecture design in order to provide more I/O parallelism inside SSDs.

There are several works that investigate in database query processing techniques on flash-based SSD [9, 10, 11, 12]. Graefe et al. [10], [11] focus on data structures and algorithms that leverage fast random reads to speed up selection, projection and join operations in query processing. They explore the impact of new page layouts on SSDs and propose FlashScan, FlashJoin and RARE-join algorithms. Another typical work is DigestJoin [12] which focuses on exploring the possibility of further improving non-index join algorithms by reducing intermediate results and utilizing fast random reads of SSDs. Different from previous studies, we try to optimize scan and join performance by exploiting internal parallelism of flash-based SSDs.

3 Internal Parallelism of SSD

In this section, we introduce internal parallel architecture of SSDs at first and then discuss how to utilize this rich internal parallelism and why we should detect SSD internals.

3.1 Internal Parallel Architecture

A flash-based SSD internal architecture is presented in Fig. 1. The *host interface* is used to connect with the host and its common interface type is SATA. *SSD controller* is the brain of an SSD which is in charge of executing I/O requests and issues commands to flash memory packages via the *flash controller*. Inside SSD, there is a *RAM buffer* to hold the mapping table and other metadata. A flash-based SSD implements internal parallelism by adopting multiple channels which be shared by a set of flash memory packages. Each channel can be operated independently and simultaneously while operations on flash memory packages attached to the same channel can also be interleaved, so the bus utilization can be optimized [5, 13]. By examining the internal architecture of SSDs, we can find that there are two typical levels of parallelism which are channel-level parallelism and package-level parallelism. Such rich internal parallelism provides us an opportunity to improve the performance of applications on SSDs.

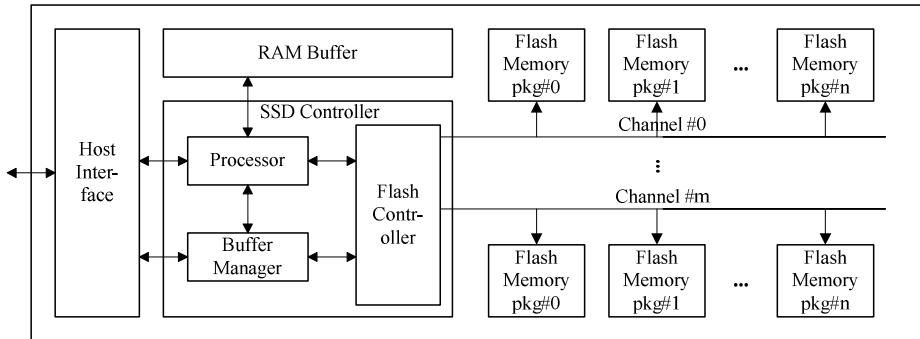


Fig. 1. Flash-based SSD internal architecture

3.2 How to Utilize Internal Parallelism

In order to utilize channel-level parallelism and package-level parallelism, multiple I/O requests designated to different flash memory packages spanning several channels should be submitted to SSDs at the same time whenever possible. In this way, native command queuing (NCQ) mechanisms of the host interface can generate favorable I/O patterns to the internal architecture [14]. In this paper, we mainly use multi-thread processing technique to produce multiple parallel I/Os at the same moment.

3.3 Detecting SSD Internals

To make better use of internal parallelism of SSDs, it is necessary to know some key architectural features of an SSD. For example, knowing the number of channels in an SSD, we can set a proper concurrency level and avoid over-parallelization. However, it is hard to obtain such key architectural information because these details are often regarded as critical intellectual property of SSD manufacturers. Therefore, we have to do some detecting work to know about the SSD internals.

We select two representative and state-of-the-art SSDs for research. One is built on multi-level cell (MLC) flash memories, which is designed for the mass storage market, while the other is a high-end product built on faster and more durable single-level cell (SLC) flash memories. For simplicity, we refer to these two SSDs as SSD-S and SSD-M respectively. More details about these two SSDs are shown in Table 1.

Table 1. Specification of Observed SSDs

	SSD-S	SSD-M
Manufacturer	Intel	Intel
Flash Memory	SLC	MLC
Capacity (GB)	32	160
Page Size (KB)	4	4
Interface Type	SATA	SATA
NCQ	32	32

Despite various implementations, most designers of SSDs try to optimize performance essentially in a similar way that is evenly distributing data accesses to maximize resource usage. Base on some open documents, Chen et al. [2] define an abstract model to characterize the internal organization of SSD. In that model, a *domain* is a set of flash memories that share a specific set of resources (e.g. channels). A domain can be further partitioned into *sub-domains* (e.g. packages). A *chunk* is a unit of data that is continuously allocated within one domain. Chunks are interleavingly placed over a number of domains. Guided by the model and the detecting method introduced in [2], we detected the chunk size and the number of domains on SSD-S and SSD-M respectively. The detecting results are shown in Table 2.

Table 2. Detecting Results

	SSD-S	SSD-M
Chunk Size (KB)	4	16
Domains Number	20	20

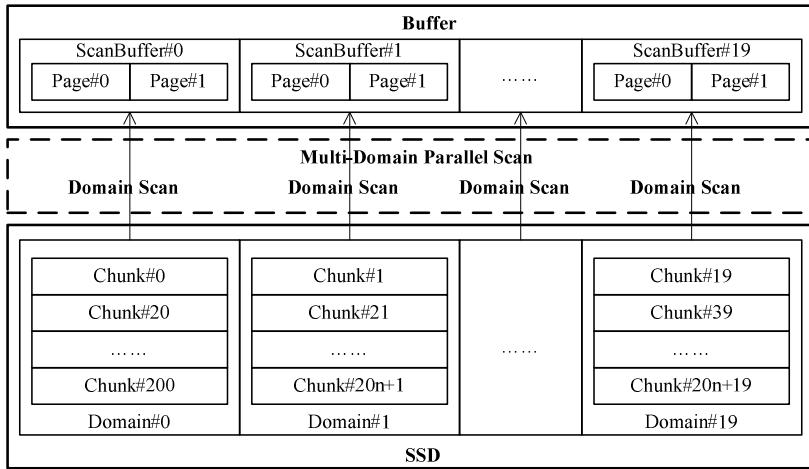
4 ParaScan

In this section, we first give an overview of ParaScan, a parallel table scan operator, which is designed guided by internal parallelism of flash-based SSDs and then describe its main components including domain scan and multi-domain parallel scan in the following subsections.

4.1 ParaScan Overview

Despite various implementations, most SSDs adopt a RAID-0 like striping data storage mechanism as is shown in Fig. 2. The striped chunks of the domains are mostly placed in consecutive LBA (Logical Block Address) regions. In the striped domains, the write interleaving technique enhances the write performance by avoiding the shared data-bus channel competition and by interleaving data transfers while other domains are writing the already transferred data. This data storage policy provides us a chance to do parallel table scan and maximize resource usage by distributing data accesses to different domains. Therefore based on that good nature, we propose a parallel table scan called ParaScan to improve the efficiency of table scan.

As shown in Fig. 2, the basic operation of ParaScan is *domain scan*. Domain scan read data chunks one by one from a single domain and then put them into the buffer. To reduce the potential performance loss caused by sharing resources, each domain scan maintains a small scan buffer called ScanBuffer. In this way, multiple domain scan can be executed in parallel without any interference. Multiple domain scan and multiple ScanBuffers compose the *multi-domain parallel scan*. As domain is a parallel unit of solid state drive, when we want to write data to SSDs, we should consider that all data pages of a relational table should be distributed into different domains as much as possible to make full use of internal parallelism of SSDs.

**Fig. 2.** Overview of ParaScan

4.2 Domain Scan and Multi-domain Parallel Scan

Domain scan is similar to traditional database table scan. Traditional database table scan read data blocks of a relational table one by one, and then put them into a buffer for processing. The different between domain scan and traditional table scan is that domain scan read data chunks one by one which all come from the same domain, and then put them into ScanBuffer. Due to the size of data stored in the same domain is usually exceed ScanBuffer size, the processed contents need to be replaced in order to read unprocessed data pages. Because here we do not consider the data write operation, the management of the ScanBuffer becomes relatively easier, we only need to cover the processed contents directly to read unprocessed data.

We implement multi-domain parallel scan by multi-thread processing. ParaScan operator generates multiple threads to do scan operation and each of them is in charge of one or multiple domain scan. Entire scan buffer is also divided into several ScanBuffers so that each scan thread can use one ScanBuffer. The performance of multi-domain parallel scan depends on the concurrency level which is not only related to the number of domains but also the maximal number of physical threads supported by the processor and the maximal queue depth supported by the SSD.

5 ParaHashJoin

Based on ParaScan operator, we present ParaHashJoin in this section. We first give an overview of our join operator in Section 5.1 and then describe its main components including ParaHash, MiniJoin and Fetch in the following subsections.

5.1 ParaHashJoin Overview

ParaHashJoin is a parallel hash join operator tuned for solid state drives. First, it parallels the join operation as much as possible to make use of internal parallelism of SSDs. Moreover, ParaHashJoin also learns some important ideas from previous researches to take advantage of the fast random reads of SSDs. It uses late materialization strategy to avoid processing unneeded attributes and postpone retrieving projected attributes until absolutely necessary. In this paper, we mainly consider a two-way equi-join implement of ParaHashJoin, and the multi-way join implement will continue to be researched in the future.

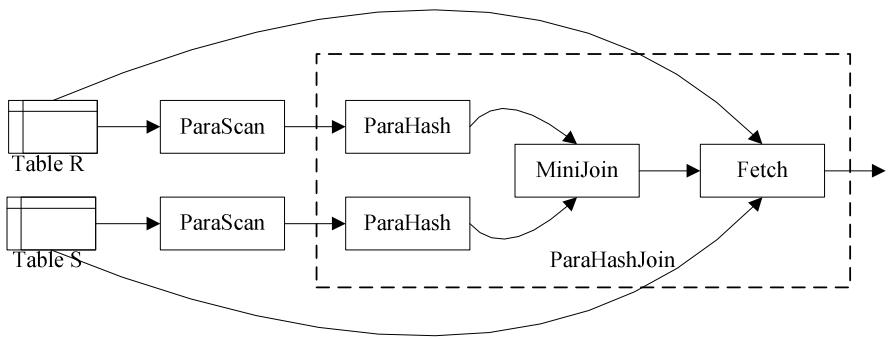


Fig. 3. Overview of ParaHashJoin

Each ParaHashJoin in two-way equi-join consists of three phases, ParaHash phase, MiniJoin phase and Fetch phase, as shown in Fig. 3. The ParaHash phase builds a hash table in parallel on the join attributes and the row-ids (RIDs) of the participating rows from input relational table. A RID specifies the page and offset within the page for that row. ParaHashJoin uses a late materialization strategy in which the Fetch phase retrieves the needed attributes using the RIDs specified in join results produced by MiniJoin. This approach offers some important benefits over traditional joins which use an early materialization strategy.

First, based on ParaScan, ParaHashJoin can hash table records belong to different domains in parallel which not only make full use of internal parallelism of SSDs but also the ability of multi-core processor. Second, ParaHashJoin is more memory efficient. By using late materialization strategy, ParaHashJoin greatly reduces the amount of data needed to be read to compute the join result. Moreover, when multiple passes are needed, it incurs lower partitioning cost than traditional joins.

However, to get these benefits, we have to pay more CPU cost to do multi-thread processing and more cost of random reads for retrieving the other projected attributes in the fetch phase. But we show that this tradeoff is worthwhile to do a join on SSDs in the experimental section.

5.2 ParaHash

In ParaHash phase, we use the same technique as ParaScan to implement parallel processing. Fig. 4 is a sketch of ParaHash in which buffer is divided into two regions, scan area and hash area. After ParaScan a table, we generate multiple hash threads to calculate the hash values of records that have been scanned into ScanBuffers, and then put their join attributes and RIDs into corresponding hash buckets in parallel. Each hash thread is in charge of one ScanBuffer. To mitigate the cost of calculations, here we use a simple hash function which applies a fast bit operator, as shown in Eq. (1). In this equation, $join_attr$ represent the join attribute value of a record to be hashed and B is the number of hash buckets which should be the power of 2.

$$\text{hash_value} = \text{join_attr} \& (B - 1) \quad (1)$$

As we implement ParaHash by multi-thread processing, some threads may hash different records into the same bucket at the same time. Thus, each bucket should maintain a lightweight lock to do concurrency control. Moreover, we build a bitmap for each bucket to quickly judge whether hash index records with specified join attribute exist in the bucket.

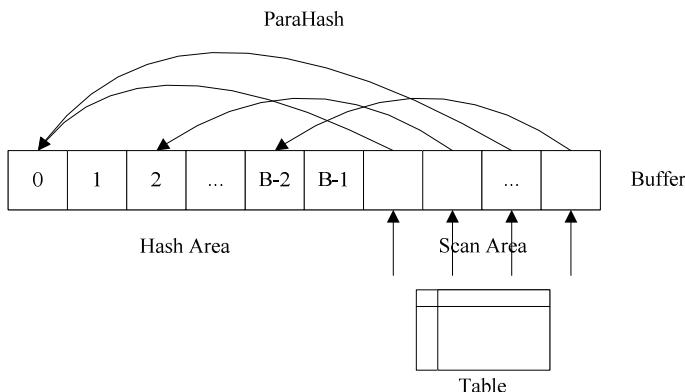


Fig. 4. Sketch of ParaHash

5.3 MiniJoin

After ParaHash table R and ParaHash table S, we have already get two hash tables. Then, in the MiniJoin phase, we read each hash bucket into memory and generate the join results, in the form of $\{join_attr, RID_R, RID_S\}$. The MiniJoin results may be written to the SSD sequentially if it is larger than the memory size.

In the case mentioned above, we need two pass to generate the MiniJoin results, one pass for ParaScan and one pass for MiniJoin. But if there is enough memory to hold the hash table of table R, the smaller table, we only need one simple pass. It first ParaScan and ParaHash table R, then it ParaScan table S. After that, in the MiniJoin phase, it can directly probe the hash table and produce MiniJoin results.

5.4 Fetch

However, the outputs of the MiniJoin phase are the incomplete join results which only tell us which records satisfy the join. Therefore, in the Fetch phase, we should fetch the necessary attributes using the RIDs specified in the join index to generate the final join results.

In this phase, an efficient fetching strategy is very important for ParaHashJoin as it minimizes the number of page accesses when fetching the needed attributes from the original tables. A straightforward strategy is to fetch the related pages specified by RIDs as soon as they are produced in the MiniJoin phase. For each join result of MiniJoin, ParaHashJoin fetch the needed data pages in the buffer pool or retrieves them from peripheral storage, and then generate the final join result. This approach is reasonable when all data pages needed to generate the result can fit in memory. However, when available memory is insufficient, this approach may result in reading some pages multiple times because the RIDs in the join index are usually unordered. Thus, the larger the join result, the higher is the cost of reloading pages. TID hash joins [15] use this approach which is their biggest weakness.

We adopt a sort-based fetching strategy inspired by DigestJoin [11] to avoid reloading pages as much as possible. Before fetching the matching pages according to RIDs in MiniJoin results, we sort MiniJoin results based on the RIDs of outer table at first. Then we begin to load needed pages to produce final join results according to the sorted MiniJoin results. In this strategy, we need to pay more cost to execute this sort, but we show that this payment is worthwhile in the experimental section.

6 Experiment Evaluation

In this section, we present experimental evaluations of ParaScan and ParaHashJoin. We first describe the experimental setup in Section 6.1. Then we present experimental results of ParaScan and ParaHashJoin on TPC-H datasets in Section 6.2 and Section 6.3 respectively.

6.1 Experimental Setup

Our experiments all run on a HP PC with Ubuntu 12.10 operating system. This platform is equipped with Intel Core i5-2400 @ 3.10GHz processor which is of four cores and supports four physical threads. In addition, it is equipped with 8G DDR3 memory, a 500G 7400rpm SATA3 Seagate magnetic disk and two kinds of SSDs as shown in Table 1. In order to make use of SSDs' rich internal parallelism, we need to enable AHCI (Advanced Host Controller Interface) mode by setting the BIOS.

For our experiments, we implement two operators, ParaScan and ParaHashJoin, by multi-thread C programming. To avoid the interference from the buffer of file system, we read/write files in DirectIO mode and align the memory manually. The test dataset is taken from the TPC-H benchmark. In particular, we use CUSTOMER table and ORDERS table to do scan and join. CUSTOMER table has 1.5 million rows in total size of about 256MB, while ORDERS table has 150 million rows in total size of about 2GB. In following subsections, we have executed scan operations on ORDERS table and join operations between CUSTOMER table and ORDERS table.

6.2 ParaScan Evaluation

We executed a table scan on ORDERS table by using ParaScan in our HDD, and two kinds of SSDs respectively. Fig. 5 compares the performance of scans as we vary the number of parallel scan threads. For simplicity, we define the symbol ParaScan-N to represent running ParaScan with N parallel threads. As traditional table scan reads data block one by one sequentially in a single-thread mode, it is equal to ParaScan-1. As our CPU is of 4 physical threads and we have known that SSD-S and SSD-M both have 20 domains, we tested the performance of ParaScan under 1, 4, 10, 20 and 30 parallel threads.

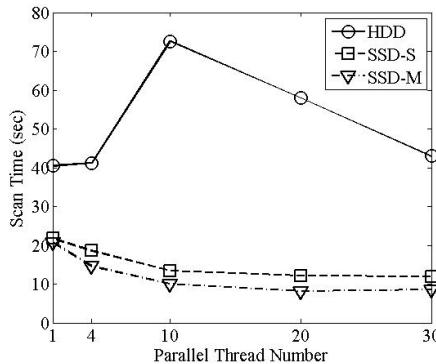


Fig. 5. Performance comparation of traditional table scan and ParaScan

According to the results shown in Fig. 5, we learn that ParaScan doesn't suit for HDD due to its magnetic characteristics. However, the scan time of ParaScan on two kinds of SSDs both decrease apparently with the increasing number of parallel threads. When the number increases to 30, the performance of ParaScan on SSDs still has an improvement but this improvement is very little. That is because the cost to switch among parallel threads also increases. Fig. 5 shows that, in best case, the scan time of ParaScan on SSD is only half of time of traditional table scan on SSD and only a quarter of time of the traditional table scan on HDD.

6.3 ParaHashJoin Evaluation

In our experiments, we compare the performance of ParaHashJoin with two kinds of join algorithms. One is TradHashJoin which refers to traditional hash join using table scan and the other one is NewHashJoin which represents traditional hash join using ParaScan. We mainly consider the equi-join of two relations CUSTOMER and ORDERS on a single join attribute and use *selectivity* to refer to the percentage of results size. The number of parallel threads in ParaScan and ParaHashJoin is set to be 20. Due to the length limit of this paper, we only present results of experiments on SSD-S but we also conducted experiments on SSD-M and got consistent results.

In the first experiment, we vary the selectivity of the join result from 0.01% (1500 rows) to 10% (1.5 million rows). The amount of memory allocated to each join is 8MB so that all of the joins are executed in two pass. As shown in Fig. 6, the execution times of all algorithms increase with the growth of the selectivity. That is because higher selectivity can lead to larger intermediate results and hence higher partitioning costs. However, ParaHashJoin suffers the least when join selectivity is small since it only partitions the join attributes and take full advantage of internal parallelism of SSD at the same time.

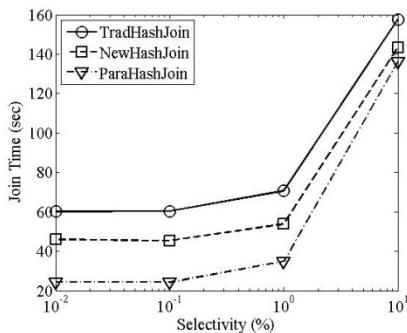


Fig. 6. Performance comparison of join operators under different selectivity

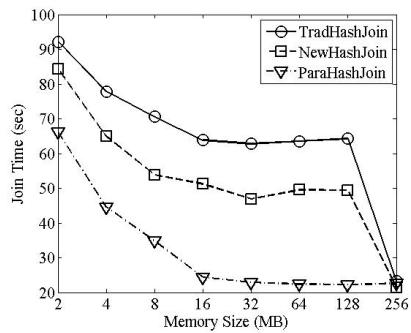


Fig. 7. Memory impact on joins

To evaluate memory impact on joins, we set join selectivity at 1% and then vary the amount of memory allocated per join from 2MB to 256MB. Fig. 7 presents our results. We can see that both TradHashJoin and NewHashJoin require 256MB to execute the join between CUSTOMER and ORDERS in one pass while ParaHashJoin only requires 16MB. Fig. 7 shows that, in best case, ParaHashJoin is 1X faster than NewHashJoin and 1.5X faster than TradHashJoin. From these results, we can see that it is worthwhile to pay extra CPU cost for multi-thread processing and extra cost of random reads for retrieving projected attributes in the fetch phase.

7 Conclusion

In this paper, we detect and examine internal architecture of different kinds of SSDs and then based on rich internal parallelism of SSDs, we propose ParaScan and ParaHashJoin operators to accelerate scan and join processing on SSDs. ParaScan takes full advantage of internal parallelism of SSDs through multi-thread processing techniques. Based on ParaScan, ParaHashJoin not only parallels the join operation as much as possible but also applies some important ideas from previous researches to take advantage of the fast random reads of SSDs. The experimental results show that ParaScan operator is 1X faster than traditional table scan on SSD and 2X faster than traditional table scan on HDD in best case. And the execution of traditional hash join algorithm on SSD is 3 times longer than ParaHashJoin. These fully demonstrate the superiority of ParaScan and ParaHashJoin.

Acknowledgements. This research was partially supported by the grants from the Natural Science Foundation of China (No. 60833005, 61070055, 91024032, 91124001); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University (No. 11XNL010).

References

1. Chen, F., Koufaty, D.A., Zhang, X.: Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In: SIGMETRICS, pp. 181–192 (2009)
2. Chen, F., Lee, R., Zhang, X.: Essential Roles of Exploiting Parallelism of Flash Memory based Solid State Drives in High-Speed Data Processing. In: HPCA, pp. 266–277 (2011)
3. Lee, S.-W., Moon, B.: Design of flash-based DBMS: An in-page logging approach. In: SIGMOD, pp. 55–66 (2007)
4. Ramakrishnan, R., Gehrke, J.: Database Management Systems, 3rd edn. McGraw Hill (2002)
5. Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J.D., Manasse, M., Panigrahy, R.: Design Tradeoffs for SSD Performance. In: USENIX, pp. 57–70 (2008)
6. Roh, H., Park, S., Kim, S., Shin, M., Lee, S.-W.: B+-tree index optimization by exploiting internal parallelism of flash-based solid state drives. Proceedings of the Very Large Data Base (VLDB) Endowment 5(4), 286–297 (2012)
7. Hu, Y., Jiang, H., Feng, D., Tian, L., Luo, H., Zhang, S.P.: Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity. In: ICS, pp. 96–107 (2011)
8. Park, S.Y., Seo, E., Shin, J.Y., Maeng, S., Lee, J.: Exploiting internal parallelism of flash-based SSDs. Computer Architecture Letters 9(1), 9–12 (2010)
9. Do, J., Patel, J.M.: Join processing for flash SSDs: remembering past lessons. In: DaMoN, pp. 1–8 (2009)
10. Shah, M.A., Harizopoulos, S., Wiener, J.L., Graefe, G.: Fast scans and joins using flash drives. In: DaMoN, pp. 17–24 (2008)
11. Tsirogiannis, D., Harizopoulos, S., Shah, M.A., Wiener, J.L., Graefe, G.: Query processing techniques for solid state drives. In: SIGMOD, pp. 59–72 (2009)
12. Li, Y., On, S.T., Xu, J., Choi, B., Hu, H.: DigestJoin: Exploiting Fast Random Reads for Flash-based Joins. In: MDM, pp. 152–161 (2009)
13. Dirik, C., Jacob, B.: The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device, architecture, and system organization. In: ISCA, pp. 279–289 (2009)
14. Lee, S.-W., Moon, B., Park, C.: Advances in flash memory SSD technology for enterprise database applications. In: SIGMOD, pp. 863–870 (2009)
15. Marek, R., Rahm, E.: TID hash joins. In: CIKM, pp. 42–49 (1994)

MixSL: An Efficient Transaction Recovery Model in Flash-Based DBMS

Yulei Fan and Xiaofeng Meng

School of Information, Renmin University of China, Beijing, China
`{fy1815,xfmeng}@ruc.edu.cn`

Abstract. With the development of flash technologies, flash disks have become an alternative to hard disk as external storage media. Because of the unique characteristics of flash disks such as fast random read access and out-place update, shadow paging technology can be adopted to support transaction recovery in flash-based DBMS. Inspired by shadow paging and logging, we propose a new transaction commit model named MixSL which can be used in databases built on MLC flash disks. Based on MixSL, we detail normal processing, garbage collection and recovery. For improving system performance and raising the utilization ratio of flash disks, we extend MixSL to support group commit. Our performance evaluation based on the TPC-C benchmark shows that MixSL outperforms the state-of-the-art recovery protocols.

Keywords: Flash Memory, Recovery, Database, Shadow Page.

1 Introduction

With the development of semiconductor technologies, flash disks have been a competitive alternative to traditional magnetic disks as external storage media in portable devices, new-generation laptops and enterprise servers. Flash disks have the unique characteristics such as fast random access, low power consumption, high shock resistance, small dimensions and light weight[1]. In fact, single-level-cell (SLC) flash disks and multiple-level-cell (MLC) flash disks have been two popular flash devices as external storage. SLC flash disks have a favorable characteristic of partial page programming, but their capacity are usually less than MLC flash disks because of the density of single bit per cell, which leads to decrease in the unit of read, write and erase operations. This leads to high production costs, so SLC flash disks are very expensive. MLC flash disks have the preponderance of price and capacity because of lower production costs and higher density of multiple bits per cell. But the lifetime of MLC flash disks is less than SLC flash disks. In this paper, we contribute to flash-based DBMSs built on MLC flash disks by investigating how transaction recovery can be supported to improve performance and raise the utilization ratio of flash disks.

As a vital part of DBMSs, transaction recovery guarantees atomicity and durability of transactions. There are two predominant approaches to support transaction recovery for DBMSs, namely write ahead logging (WAL)[2] and

shadow paging[3]. For WAL based on in-place update, in addition to undo/redo log records, we need some special log records to record significant events during transaction processing. For short transaction, WAL needs frequent write operations which are not preferable on flash disks. But group commit[3] can be used to improve the performance of transaction recovery. Different from WAL, shadow paging adopts out-of-place update style which is good suit for flash memory because of the erase-before-write limitation. In shadow paging, a page mapping table mapping page IDs to disk addresses is the key to access data, which is an indispensable component in flash translation layer(FTL)[4] of flash disks. Because of the unique characteristic of flash disks, some issues, such as maintaining the mapping between logical addresses and physical addresses, no longer exist for flash disks. These render shadow paging an appealing solution to support efficient transaction recovery on flash disks.

In this paper, based on shadow paging and logging, we propose a new commit scheme, named MixSL, for supporting efficient transaction recovery in flash-based DBMSs built on MLC flash disks. Then we detail normal processing, garbage collection and recovery. Simultaneously we extend MixSL to group commit for the system performance and the utilization ratio of MLC flash disks. On the whole, the contributions of this paper are summarized as follows:

- We propose a new commit model, called MixSL, assisting shadow paging with logging, which supports efficient transaction recovery in flash-based DBMSs built on MLC flash disks.
- Based on MixSL, we detail normal transaction processing including updating data pages, committing and aborting transactions. And then we present the garbage collection and recovery algorithms.
- We extend MixSL to support group commit for improving transaction processing performance and the utilization ratio of flash disks.
- We conduct a performance evaluation of MixSL based on the TPC-C benchmark.

The rest of this paper is organized as follows. Section 2 introduces the background and related work of our research, including flash disks and flash-based recovery approaches. In the section 3, we present MixSL and normal transaction processing, garbage collection, and recovery. Section 4 discusses how to extend MixSL to support group commit. We present the performance evaluation results of MixSL in section 5. Finally, we conclude this paper and discuss future directions in section 6.

2 Background and Related Work

In this section, we describe the relevant characteristics of MLC flash disks as well as the flash translation layer (FTL). And then we present related work about flash-based transaction recovery.

2.1 Flash Characteristics

A flash disk consists of a number of flash memory chips which are organized in many blocks. Each block is composed of a fixed number of pages. A typical page consists of 2K bytes data area and 64 bytes spare area. The spare area often stores metadata such as the error correction code (ECC) and logical block address (LBA), and so on. So, a typical block size is 128K+4K bytes (i.e., 64 pages). Erasure operations must be performed at the block level, while read and write operations are performed at the page level. There are two types of flash available in the current market: NOR and NAND. NOR flash is mainly used to store the programs. NAND flash is designed as the mass storage devices and also is the focus of this paper. Moreover, according to the number of bits stored on each cell, NAND flash can be categorized into single-level cell(SLC) and multi-level cell(MLC).

Flash disks possess a number of unique I/O characteristics[5]. First, because flash disks have no any mechanical components, the latency of a request is only linearly proportional to the amount of data transferred. Second, flash memory is subject to an erase-before-write constraint. Out-place updating is often supported through flash translation layer (FTL) for addressing the erase-before-write constraint. Third, another important characteristic of flash memory is asymmetric read, write and erasure speeds, while magnetic disks have the symmetric read and write speed and there are no erase operation for hard disks. Forth, each block can survive only a limited number of erasure operations, which means that it wears out and becomes unreliable after limited number of writes on flash disks. Typically, SLC flash supports 100K erasures per flash block, and the MLC flash only supports about 10K erasures because of the higher bit density.

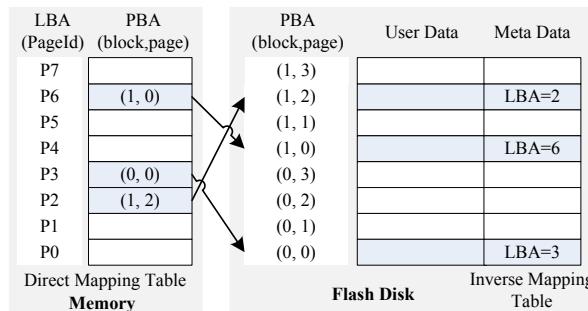


Fig. 1. FTL Mapping Table

Flash translation layer(FTL) is an important component to emulate a hard disk. FTL provides an interface to read and write flash pages and support transactional operations (page writes, commit, abort, and recovery). As shown in Figure 1, a direct mapping table in memory records the mappings between logical page addresses LBAs and physical page addresses PBAs; LBAs in the spare

areas of flash pages form an inverse mapping table, which is used to re-build the direct mapping table at boot time. The two mapping tables are crucial to implement out-place updating. Every out-of-place update operation leaves an obsolete page on the flash, so FTL maintains a free block list and has a garbage collection module to reclaim obsolete pages. Wear leveling module in FTL is used to spread writes/erasures uniformly across the entire disk space for lengthening the lifetime of flashes.

2.2 Related Work

Early works attempted to emulate the interfaces of traditional magnetic disks, so there are many work about FTL. According to mapping address granularity, FTL algorithms can be divided into four categories[6]: page-level mapping, block-level mapping, hybrid mapping with page and block granularity, and other mapping.

Recently, flash disks have been exploited to enhance the performance of file systems and database systems from various aspects. For flash-based storage management, log-based and log-structure mechanisms have been suggested to optimize random write operations by sequential write operations, such as IPL[7] and FlashStore[8], and so on. In addition, for index and buffer management, there are many new index structure[9] and new buffer replacement algorithms[10] proposed for flash devices. Simultaneously query processing techniques have also been intensively studied[11].

In contrast, not much work has been done on flash-aware transaction management. For flash-based file system, Prabhakaran et al.[12] have developed a shadow paging-based scheme, called cyclic commit. Based on this idea, they proposed two protocols, SCC and BPCC, but they are not fit for flash-based DBMSs. First, the current shadow page must be buffered until the arrival of the next shadow page for forming a cyclic linked list. Second, uncommitted pages must be frequent erased in SCC. Third, in BPCC, a complicated garbage collection mechanism need be performed. In [13], Wu proposed a fast recovery scheme for flash-based log-based file systems, so the log records are much different from log records in DBMSs. And the log records are committed into a special area on flash disks in order to read the special area during recovery.

For flash-based DBMSs built on SLC flash disks, based on shadow paging, Sai Tung On et al.[14] proposed a new flagcommit scheme for efficient transaction recovery. It addresses these issues presented in[12]. Because SLC flash disks have an unique characteristic of partial page programming, which allows a flash page to be updated a few times before an erasure becomes mandatory, so they can exploit this partial page programming feature to keep track of the transaction status. Flagcommit stores a flag about a transaction status in spare area of every flash data page. So transaction commit processing can be accomplished by updating some flags in the style of in-place. But MLC flash disks have no the unique characteristic of partial page programming. And some extra read operations need be performed for locating the data page when the transaction is aborted or committed. Moreover, during collecting the obsolete pages, some in-place operations have to be done for keeping track of the transaction status.

3 MixSL: Shadow Paging + Logging

Inspired by shadow paging and logging, we present the new commit model, MixSL, as shown in Figure 2, which exploits out-of-place update style of Flash-based SSDs and keeps track of the transaction status by logging. And then we will detail normal transaction processing, garbage collection and recovery.

3.1 Commit Model: MixSL

The basic idea of commit model MixSL is to use shadow paging to keep track of update operations, and use logging to keep track of transactions' status. Based on MixSL, there are a link and the transaction ID in each shadow page. The link pointing to the preceding version page of the shadow page, can be used for garbage collection and transaction recovery. The transaction ID is the identifier of the transaction producing the page. However, different from SLC flash disks, MLC flash disks have no unique characteristic of partial page programming in SLC flash disks, so we must keep track of the transaction status by logging. Due to good performance of writing fixed area[5], log records related to transaction commit protocol are maintained in the fixed flash area. Log records in the fixed flash area and metadata in every shadow page can be used to undo the updates and recovery during transaction rollback or system restart.

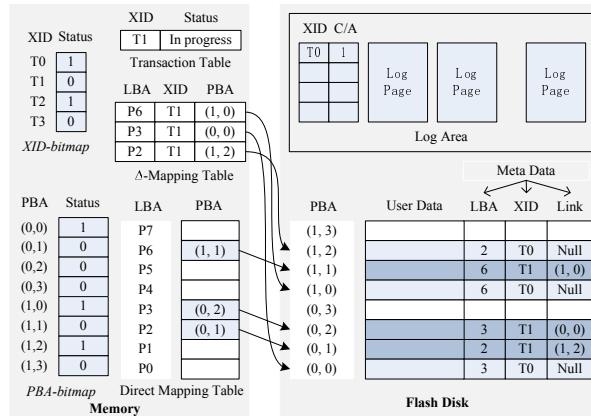


Fig. 2. Commit Model: MixSL

In traditional disk-based DBMSs, data buffer pool caches the frequently accessed disk pages of the database, and log buffer pool caches the log records produced by running transaction. The buffer management policy has an impact on the transaction recovery mechanism. In this paper, MixSL model works with steal and force policies. For log buffer management, because there are only some transaction log records but no redo/undo log records, force policy is adopted in

log buffer management. In other words, when a transaction commit, log buffer page must be flushed into the fixed flash area of flash disks. For data buffer management, we need make some assumptions. First, a page-level concurrency control protocol is adopted to handle update conflicts. Second, we used a write-back buffer: a shadow page is not created on the flash disk until the page is evicted from the buffer pool or the corresponding transaction commits. For improving the write performance of DBMSs and the lifetime of flash disks, so in section 4, we extend MixSL to support group commit.

To complete normal transaction processing, garbage collection and recovery, some data structures should be maintained in memory as shown in Figure 2. ***Transaction Table*** stores each transaction ID XID and its status $Status$ (i.e., in progress, committed, or aborted). **Δ -mapping Table** stores all in-progress transactions IDs XID as well as the logical-to-physical mappings $\langle LBA, PBA \rangle$ produced by them. ***Direct mapping Table*** maintained the newest committed logical-to-physical mappings $\langle LBA, PBA \rangle$. For each transaction, there is a bit in ***XID-bitmap*** (aborted transaction: '0' (default value), committed transaction: '1'). For each physical page, there is a bit in ***PBA-bitmap*** (valid page: '1' (default value), invalid page: '0'). XID-bitmap and PBA-bitmap are only built during garbage collection and recovery.

As shown in Figure 2, the whole flash disk space is divided into two regions: ***Data Area*** and ***Log Area***. Every flash data page consists of a data area and a spare area. Data area stores users' data, but spare area stores some metadata including LBA , XID , $Link$. LBA is the logical page ID of this physical flash data page. XID is the ID of the transaction T producing this data page. $Link$ points to the preceding version data page (NULL if it is the first version data page), by which all obsolete data pages are chained together. Log area stores all log records keeping track of all transactions' final status. Every log record consists of XID and C/A . XID stores a transaction ID. C/A represents the transaction's final status ('1' if transaction is committed; otherwise, '0').

3.2 Normal Processing

Based on MixSL, we detailed normal transaction processing including updating a data page on flash disk, committing a transaction and aborting a transaction.

Update. When a transaction T^* with the ID xid updates a logical page lp with the logical address lpa , MixSL performs the following three steps. First, if the transaction T^* doesn't have existed in transaction table, the transaction T^* and its status *in-progress* are inserted into it. Second, we create a shadow page pp' on flash disks with the physical address ppa' , and simultaneously filling in the LBA, Link and XID fields in the spare area of pp' with lpa , ppa' and xid . Last, an entry $\langle lpa, T^*, ppa' \rangle$ is inserted into Δ -mapping table. It is used to complete the transaction T^* and guarantee the correctness of direct mapping table.

Commit. When a transaction T^* with the ID xid commits, MixSL performs the following three steps. First, a log record $\langle T^*, 1 \rangle$ is inserted into log buffer and then flushed into fixed flash area. Second, all entries $\langle lpa, T^*, ppa \rangle$ of

the transaction T^* in Δ -mapping table are merged into direct mapping table to replace entries $\langle lpa, ppa \rangle$ with the same lpa . Simultaneously these entries $\langle lpa, T^*, ppa \rangle$ of the transaction T^* need be deleted from Δ -mapping table. Last, transaction T^* must remove from transaction table.

Abort. When a transaction T^* with the ID xid aborts, MixSL performs the following three steps. First, a log record $\langle T^*, 0 \rangle$ is inserted into log buffer and then flushed into fixed flash area. Second, all entries same as $\langle -, T^*, - \rangle$ of the transaction T^* must be deleted from Δ -mapping table. Last, transaction T^* must remove from transaction table.

3.3 Garbage Collection

When the amount of free space on flash disks becomes lower than some pre-set threshold, garbage collection is triggered to reclaim the obsolete pages. The obsolete pages include three following types: (1) the uncommitted page; or (2) the aborted page; or (3) the committed out-of-date page. The transaction status becomes critical on identifying the 1st and 2nd kinds of pages, so we need create

Algorithm 1. Garbage Collection

```

begin
  Initialize_XID_Bitmap(0);
  Initialize_PBA_Bitmap(1);
  for each page in fixed log area do
    for each record in the page do
      xid = XID field of log record;
      stat = Status field of log record;
      if stat == 1 then
        Set_XID_Bitmap(xid, 1);

  for each page on flash disks do
    xid = XID field in the spare area;
    lpa = LBA field in the spare area;
    ppa = Physical address of the data page;
    prelink = Link field in the spare area;
    stat = Get_XID_Bitmap(xid);
    if stat == 0 then
      Set_PBA_Bitmap(ppa, 0);
    else
      Set_PBA_Bitmap(prelink, 0);

  for each bit in PBA_bitmap do
    flag = the bit value;
    if flag == 0 then
      ppa = TransferLocation();
      Free_Page(ppa);
      Add_Free_Block_List(block_num);

end

```

XID-bitmap. The transaction status in XID-bitmap and *Link* field in spare area of flash pages are combined for identifying the 3rd kind of page. So the garbage collection is shown in algorithm 1.

The first step is to create and initialize XID-bitmap and PBA-bitmap with 0 and 1 respectively. The first two *For* loops set XID-bitmap by reading log records. The 3rd *For* loop reads and processes data pages on flash disks one by one. If the transaction producing a data page is uncommitted or aborted, the data page can be collected, i.e., it is invalid. So the bit corresponding to the data page in PBA-bitmap is set as 0. If the transaction is committed, we can make sure that pre-version of the data page is invalid, so the bit corresponding to *Link* in PBA-bitmap is set as 0. By now, we identify whether every data page is valid or invalid, so we can perform garbage collection for every block including invalid data pages in the last *For* loop. During collecting garbage, we need change the direct mapping table because valid data pages need be moved. At last, the erased blocks need be inserted into free block list.

3.4 Recovery

After a normal shutdown or system failure, a recovery procedure is invoked when the system restarts. It recovers the newest committed version data page and rebuilds the direct mapping table. As same as garbage collection, we need identify whether every data page on flash disks is valid or invalid. So the first three steps of recovery procedure are very like these of garbage collection as shown in algorithm 2.

The first 3 *For* loops are same as these in algorithm 1, so the status (i.e., valid or invalid) of every data page can be known by getting every bit value of PBA-bitmap. In the last *For* loop, we random read valid data pages, and then gain their *LBA* fields in spare area, and insert $\langle lpa, ppa \rangle$ into direct mapping table. Recovery is only related to random read, because of good random read

Algorithm 2. Recovery

```

begin
    Initialize_XID_Bitmap(0);
    Initialize_PBA_Bitmap(1);
    //The first 3 For loops are same as these in algorithm 1;
    page_num = 0;
    for each bit in PBA_bitmap do
        flag = Get_PBA_Bitmap(page_num);
        if flag == 1 then
            Random_Read_Page(page_num);
            lpa = LBA field in the spare area;
            ppa = Physical address of the data page;
            Insert_Direct_Mapping_Table(lpa, ppa);
        page_num = page_num + 1;
end

```

performance of flash memory, so the performance of recovery is only determined by the number of needed pages.

4 Extended MixSL: Group Commit

In this section, we extend MixSL to support group commit for improving the performance of flash-based databases and advancing the utilization ratio of flash space. Because a force buffer policy is adopted by log buffer, for every transaction, if it commits or aborts, we must immediately flush log buffer page into flash disk. But this may bring several problems, first, it leads to frequent write operations. Second, the flushed log page can be not full, so the flash space is dissipated. Because the unit of read and write operations is page, the log buffer page can not be smaller than one flash page. Group commit means that some transactions can be committed at the same time. So some transactions must be postponed committing or aborting. There are two protocols based on group commit. The first is to group commit fixed number of transactions, where the number can be specified by user according the buffer size and the application system. The second is to group commit dynamic number of transactions, which is more suitable to a complex application system. For the second protocol, we can group commit transactions when all the log records produced by these transactions can be fill in one log buffer page.

5 Performance Evaluation

In this section, we evaluate the performance of our proposed MixSL based on the TPC-C benchmark. Firstly, we describe the experiment setup for simulating MixSL and FlagCommit [14]. And then we compare MixSL with the CFC and AFC protocols in FlagCommit.

5.1 Experiment Setup

For showing performance, we implement a trace-driven SSD simulator which can simulate SLC SSD and MLC SSD by configuring its parameters. For comparing the performance of FlagCommit and MixSL, we configured the simulator to emulate a 32GB SLC SSD and a 32G MLC SSD with a wear-aware garbage collection. Similar to [12], 10% of the flash blocks were reserved for handling garbage collection, and the threshold to trigger the garbage collection was set to 5%. And then we implemented MixSL, CFC and AFC protocols based on FlagCommit with a Strict Two-Phase Locking (2PL) protocol. For the buffer management, LRU strategy is applied to cache previously disk pages accessed. The on-line transaction processing workload, TPC-C benchmark, is used for evaluating the performance of MixSL and FlagCommit. We generated the workload trace by executing TPC-C transactions on PostgreSQL 8.4 and recording their data access requests. For the generator, we set 50 clients and 20 data warehouses(2G

Table 1. Default Parameter Settings

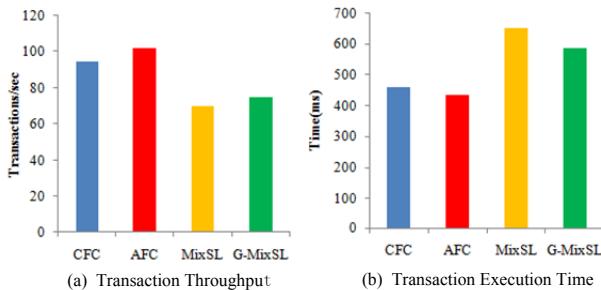
Parameter	Default Setting
SLC Block/Page size	128KB/2KB
SLC Page write/read latency	0.2ms/0.08ms
SLC Partial programming latency	0.2ms
SLC Block erasure delay	1.5ms
MLC Block/Page size	512KB/8KB
MLC Page write/read latency	0.65ms/0.25ms
MLC Focus programming latency	0.5ms
MLC Block erasure delay	5ms
Logical page size	8KB
Buffer pool size	512

data). Simultaneously, the transaction abort ratio was set to 5% by default. As shown in [14], the cost of partial programming was also same as that of a page write. So we summarize the default parameter settings in table 1.

We conducted our experiments on a HP computer running Ubuntu Linux with an Intel Xeon E5620 2.40 GHz cpu. We measured the transaction throughput, transaction execution time, recovery cost, and garbage collection overhead.

5.2 Comparison with FlagCommit

We compare the proposed MixSL with CFC and AFC protocols [14].

**Fig. 3.** Transaction Throughput and Execution Time

Transaction Throughput and Execution Time. As shown in Figure 3(a), transaction throughput of MixSL does not outperform these of CFC and AFC. Simultaneously the average running time of every transaction in MixSL is bigger than these in CFC and AFC. The main reason is that read/write latency of SLC flash is less than that of MLC flash. But the gap of transaction processing performance between SLC flash and MLC flash is less than the gap of read/write performance between them because page size of MLC is more than that of SLC.

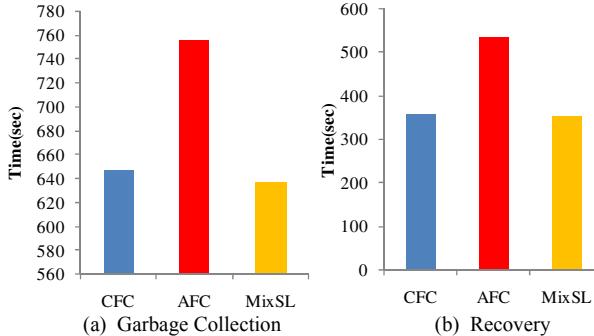


Fig. 4. Recovery and Garbage Collection

Recovery and Garbage Collection. To gain more insight, we further measure their garbage collection overhead and plot the results in Figure 4(a). MixSL outperforms AFC because garbage collection in AFC need read some pages for locating the first valid page and marks the page when the list of pages belonging one transaction is split. we can also see that MixSL is same as CFC, because of good performance of read/write of SLC flash and fast locating the page which need be marked in CFC. Figure 4(b) shows the recovery performance results which is similar with garbage collection because the part procedure of recovery is same as part of garbage collection.

Group Commit. Because of group committing transactions, flushing transaction log records into fixed area of flash memory is decreased. Group committing transactions leads to bring down the average running time of every transaction. Transaction throughput of MixSL supporting group commit (short for G-MixSL) is bigger than MixSL. So as is shown in Figure 3, the average execution time of every transaction in G-MixSL is less than that in MixSL.

6 Conclusions and Future Works

In this paper, we have proposed a new transaction commit model MixSL for database built on MLC flash memory. Our main idea is to exploit the fast random read access, out-place updating and per-page metadata to optimize the performance of transaction processing and recovery by combining shadow paging and logging technologies. Simultaneously we extend MixSL to support group commit for improving the performance of flash-based databases and advancing the utilization ratio of flash space. Our performance evaluation based on the TPC-C benchmark shows that MixSL outperforms the state-of-the-art recovery protocols. As for future work, we plan to extend MixSL to support for no-force buffer management, fine-grained concurrency control and checkpoint, and so on.

Acknowledgments. This research was partially supported by the grants from the Natural Science Foundation of China (No. 60833005, 61070055, 91024032, 91124001); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University (No. 11XNL010).

References

1. Gray, J., Fitzgerald, B.: Flash disk opportunity for server applications. *Queue* 6(4), 18–23 (2008)
2. Mohan, C., Haderle, D., Lindsay, B., Pirahesh, H., Schwarz, P.: ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Trans. Database System* 17(1), 94–162 (1992)
3. Ramakrishnan, R., Gehrke, J.: *Database Management Systems*. McGraw-Hill (2003)
4. Chung, T.-S., Park, D.-J., Park, S., Lee, D.-H., Lee, S.-W., Song, H.-J.: A survey of Flash Transalation Layer. *Journal of Systems Architecture - Embedded Systems Design(JSA)* 55(5-6), 332–343 (2009)
5. Bouganim, L., Jónsson, B.T., Bonnet, P.: uFLIP: Understanding flash IO patterns. In: *Proceedings of CIDR* (2009)
6. Ma, D., Feng, J., Li, G.: LazyFTL: A Page-level Flash Translation Layer Optimized for NAND Flash Memory. In: *Proceedings of ACM SIGMOD 2011* (2011)
7. Lee, S.-W., Moon, B.: Design of flash-based DBMS: An in-page logging approach. In: *Proceedings of ACM SIGMOD* (2007)
8. Debnath, B., Sengupta, S., Li, J.: FlashStore: High throughput persistent key-value store. In: *Proceedings of VLDB* (2010)
9. Agrawal, D., Ganesan, D., Sitaraman, R., Diao, Y.: Lazy-adaptive tree: An optimized index structure for flash devices. In: *Proceedings of VLDB* (2009)
10. Ou, Y., Härdler, T., Jin, P.: CFDC: a flash-aware replacement policy for database buffer management. In: *Proceedings of DaMoN* (2009)
11. Tsirogiannis, D., Harizopoulos, S., Shah, M.A., Wiener, J.L., Graefe, G.: Query processing techniques for solid state drives. In: *Proceedings of SIGMOD* (2009)
12. Prabhakaran, V., Rodeheffer, T.L., Zhou, L.: Transactional flash. In: *Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2008)*, San Diego, CA, USA (2008)
13. Wu, C.-H., Kuo, T.-W., Chang, L.-P.: Efficient initialization and crash recovery for log-based file systems over flash memory. In: *Proceedings of the ACM Symposium on Applied Computing*, SAC 2006 (2006)
14. On, S.T., Xu, J., Choi, B., Hu, H., He, B.: Flag Commit: Supporting Efficient Transaction Recovery in Flash-based DBMSs. *IEEE Transactions on Knowledge and Data Engineering* (2011)
15. Bernstein, P.A., Hadzilacos, V., Goodman, N.: *Concurrecy Control and Recovery in Database Systems*. Addison Wesley (1987)

Keyword Oriented Bitmap Join Index for In-Memory Analytical Processing

Yansong Zhang^{1,3}, Mingchuan Su^{1,2}, Xuan Zhou^{1,2}, Shan Wang^{1,2}, and Xue Wang^{1,2}

¹ School of Information, Renmin University of China, Beijing 100872, China

² DEKE Lab, Renmin University of China, Beijing 100872, China

³ National Survey Research Center at Renmin University of China, Beijing 100872, China
zhangys_ruc@hotmail.com

Abstract. Nowadays computers are equipped with multicore processors and large RAM to support high performance processing. In-memory analytical processing and just-in-time data warehousing have become realistic for various enterprises. An analytical query normally requires a small proportion of ‘hot’ data, usually defined by a set of keywords, instead of the entire data set, which involves large volume table scan and costly star joins. Therefore, identifying frequent keywords to retrieve hot data can dramatically reduce the cost of full table scan or star-join. In this paper, we propose a keyword oriented bitmap join index to improve the space efficiency and performance of in-memory data warehouse. Keyword oriented bitmap join index is a global bitmap join index for the entire data warehouse, as opposed to conventional bitmap join indexes which are indicated for specified attributes. With our index, star-join is first converted into keyword search and bitmap combination. The resulting bitmap filters are then employed to filter records. Through the filtering by bitmaps, a star-join is converted into positional scan on the fact table and additional dimension filtering. Both memory bandwidth and analytical performance can then be improved.

Keywords: keyword, bitmap join index, star join, hot data.

1 Introduction

With the rapid development of hardware, multicore processors and large RAM have become a common configuration for modern computers. HANA[1] proposed an in-memory computing platform with dramatic performance enhancement. It has been deployed in hundreds of enterprise applications. Oracle Exadata X3 Database In-Memory Machine[2] also employs TBs of RAM and flash memory to achieve high performance. RAM is already large enough for complete in-memory data analytics. However, in-memory computing mechanisms still need further optimization to make full use of the multicore and the large RAM.

In [3], the authors announced that memory access is a new bottleneck for in-memory computing. Column-store, the BAT (Binary Associated Table) algebra and the radix-clustering are key techniques to optimize memory bandwidth and memory

access latency. Nowadays column-store has almost become a standard feature for in-memory data warehouses. Examples include MonetDB, HANA and HyPer[4]. Analytical processing usually relies on column scan and column join to filter out undesired data. Considering the high selectivity of multidimension filtering (in SSB[5] the selectivity varies from 3.4% to 0.0000762%), column scan can actually be optimized by converting sequential scan into positional scan by skipping the useless accesses. By ‘hot data’, we refer to those which are frequently accessed by application. In data warehouses, these ‘hot data’ are usually defined by keywords in dimension tables and can be mapped to the big fact table. For example, the user group in ‘Beijing’, the product category of ‘gift’ and the date of ‘Valentines day’ are typical keywords to retrieve hot data. The useful information linked by the keywords is usually located in fact table. Therefore, if we can create a join index to identify the join relation between ‘hot’ keywords and fact records, we could improve the performance of data warehouse significantly. The concept of join index is not new. For instance, Oracle’s bitmap join index[6] is a bitmap index for the join of two or more tables. However, as a join index is very space consuming, it is usually very expensive to maintain such an index completely in memory. According to the Power law[7](also known as 80-20 rule) phenomenon, only 20% attributes are frequently accessed attributes in a database; inside the an attribute, only 20% items are frequently accessed items. Therefore, if a join index is only dedicated to the 20% of the most frequently accessed items, we can achieve 80% of the performance improvement.

In this paper, we focus on how to identify the hot data using keywords and how to set up space efficient bitmap join index for data warehouse. A data warehouse query normally looks like the following example (SSB Q4.1):

```

SELECT d_year, c_nation, SUM(lo_revenue-lo_supplycost) AS profit
FROM date, customer, supplier, part, lineorder
WHERE lo_custkey=c_custkey
    AND lo_suppkey=s_suppkey
    AND lo_partkey=p_partkey
    AND lo_orderdate=d_datekey
    AND c_region='AMERICAN'
    AND s_region='AMERICAN'
    AND (p_mfgr='MFGR#1' OR p_mfgr='MFGR#2')
GROUP BY d_year, c_nation
ORDER BY d_year, c_nation

```

The WHERE clause in the box involves the join relations between the fact and dimension tables as well as the predicates for specifying the multidimensional area being observed. We can represent the predicates with four keywords: ‘c_region_AMERICA’, ‘s_region_AMERICA’, ‘p_mfgr_MFGR#1’ and ‘p_mfgr_MFGR#2’. To facilitate the data access, we can create a bitmap index for these keywords to identify which positions in the fact table satisfying the predicate expressions. When similar queries are invoked, we can reuse the bitmap to filter fact table. If we use hot keywords as hot data axis, the hot data areas in a data warehouse can be easily retrieved by the combination of the hot keywords. Traditional index scan only filters records for routine join operations. In contrast, we first filter the fact records by keyword filters, then convert star-join operation

as piping the fact records through predicate bitmap filters on dimensions. By this approach, a query with star-join is simplified as a positional scan and filtering operation.

Our contributions can be considered as two-folds:

1. We propose a keyword oriented bitmap join index strategy to dynamically define hot data for both flexibility and space efficiency.
2. We propose to use two kinds of bitmap filters to simplify analytical processing. Keyword bitmaps are used to produce global keyword filter. Predicate bitmaps are used as star-join filter for survived records to further filter final result set.

The related work is presented in Sec.2. Keyword oriented bitmap join index is discussed in Sec.3. Sec.4 presents the results of experiments. Finally, Sec.5 summarizes the paper.

2 Related Work

2.1 Hot Data Identification

Both in OLTP and OLAP workloads, data accesses always tend to be skewed. In OLTP workloads, hot data are records with high frequency, so the key issue is to identify which records are frequently accessed by record counter or sampling method. [8] proposed a sample record access method with record accesses on log, *forward* and *backward* algorithms are designed for past-to-present scan and present-to-past scan to estimate record access frequency. [9] emphasized that recency (i.e., closeness to the present) is more important than frequency (i.e., the number of appearance), and proposed multiple bloom filter based hot data identification to capture recency.

In OLAP workloads, even hot data may involve millions of records in a big data warehouse. Moreover, data warehouse employs multidimensional model, and hot data in data warehouse is commonly defined as hypercube with predicates on different dimensions such as “c_region=‘AMERICA’ AND s_region=‘AMERICA’ AND (p_mfgr=‘MFGR#1’ OR p_mfgr=‘MFGR#2’)”; each predicate expression denotes a slice on different axis. It is simple and efficient to identify hot predicate expressions instead of identify real involved large records. Data warehouse is defined with many dimensions and hierarchies, analytical processing usually follows the pre-defined hierarchy which is represented as keywords. Hot data is defined with multiple keywords and the hot keywords also denote hot slices. So identifying hot keywords and using combined hot keywords to identify hot data cost much less than identifying hot data records.

2.2 Bitmap Join Index

We use bitmap as keyword filter. So we need a bitmap join index for hot keywords. Bitmap join index can be considered as bitmap group for high frequent dimensional attribute. Each item in the attribute has a bitmap to identify the join relation between fact table and specified dimension table on current keyword. High cardinality attributes are not suitable for bitmap join indexes for high space cost. Bitmap join

index acts as additional filter for big fact table, positions in bitmap are mapped to *rowid* for direct access instead of costly full table scan. Compared with B+-tree index, bitmap join index is small and efficient; especially that bitmap join index represents join relations between multiple tables and can reduce some costly join operations.

Index is tradeoff between space and performance. [10] proposed bitmap join index selection strategy based on cost models of bitmap join index storage, maintenance, data access and join without indexes. Bitmap join index size can be accurately computed with cardinality and rows in fact table, and selectivity in each bitmap is accurate for evaluating data access cost.

For in-memory analytical processing, original data volume is big, and RAM is expensive for high value data. So a light-weight bitmap join index is necessary for global frequent keywords instead of frequent attributes to balance both the space efficiency and performance.

3 Keyword Oriented Bitmap Join Index

3.1 Identify Hot Keywords

Keywords are extracted from **WHERE** clauses except equal join clauses no matter they are from fact table, dimension tables or which attributes they belong. Keywords are organized with uniform name, for example:

- d_year=1993 \leftrightarrow d_year_1993
- lo_discount between1 and 3 \leftrightarrow lo_discount_between_1_3
- lo_quantity < 25 \leftrightarrow lo_quantity_<_25
- s_region = 'AMERICA' \leftrightarrow s_region_AMERICA

The principle of uniform keyword name is: table name(in short)_attribute name_operator(equal can be neglected)_key(between operator has another key).

We use counter list for each keyword in which each item denotes one counter in specified time window. So we can maintain a frequency list for continuous periods, recency can also be computed with latest frequency items. We also set up a weighted frequency model based on frequency list and selectivity.

$F = a_0 + 2^{-1}a_1 + 2^{-2}a_2, \dots, 2^{-n}a_n$, where a_i is the i th frequency of time window $_i$, frequency reduces as time window get old, and the latest windows have higher effectiveness for the weighted frequency; let $F_w = (2-s)*F$, where s represents selectivity of keyword varies from 0 to 1, $(2-s)$ generates a selectivity weight(e.g. 5% for 1.95), so weighted frequency F_w can represent frequency, recency and selectivity. We use TOP K method to create K candidate keywords; K is configured according to available RAM size. Let m denotes available RAM size(byte), let n denotes record number of fact table, we can get maximal keyword candidate size $K = 8*m/n$.

3.2 Organize Keyword Oriented Bitmap Join Index

Each selected keyword needs to create a bitmap to identify which positions current keyword maps in fact table. We need a mechanism to store keyword and bitmap information, when and how to create bitmaps for specified keywords.

As keywords are not extracted from same attribute but extracted from the whole database only if they are frequently accessed in recent time and the selectivity is low enough. We use key(bitmap store for keyword oriented bitmap join index as the following example, in which we use uniform keyword name as key. Key(bitmap store can employ a hash table structure or pluggable key/value store such as Memcached[11] or Redis[12]. The key consideration is that we can efficiently locate bitmap by keyword.

c_region_America	011100001000101100001011100000...
s_region_America	010000100001000000000110100001...
c_nation_China	
s_region_Asia	000000100101100010000110100100...
p_mfgr_MFGR#1	000000100001100000000010100000...
p_mfgr_MFGR#2	
d_year_1997	

Key(bitmap store example

Keyword weighted frequency counters decide K candidate keywords. Bitmaps are created by join operation, we can attach keyword bitmap creating procedure to similar query with same keyword, and save bitmap in key(bitmap store; or we can generate all candidate keyword bitmaps in a batched processing.

3.3 Keyword Filtering for Analytical Processing

In a typical analytical query like Q4.1 in section 1, we first extract keywords from query and probe in key(bitmap store to find matched bitmaps. Figure 1 shows how to generate the keyword filter.

Keywords ‘c_region_AMERICA’, ‘s_region_AMERICA’, ‘p_mfgr_MFGR#1’ and ‘p_mfgr_MFGR#2’ are extracted from query and first three keywords have bit maps. Because ‘p_mfgr_MFGR#1’ and ‘p_mfgr_MFGR#2’ are connected with OR, and only one keyword has bitmap, so we discard keyword ‘p_mfgr_MFGR#1’ for global keyword bitmap filter. We combine keyword bitmaps of ‘c_region_AMERICA’ and ‘s_region_AMERICA’ with bit-wise AND for global keyword filter.

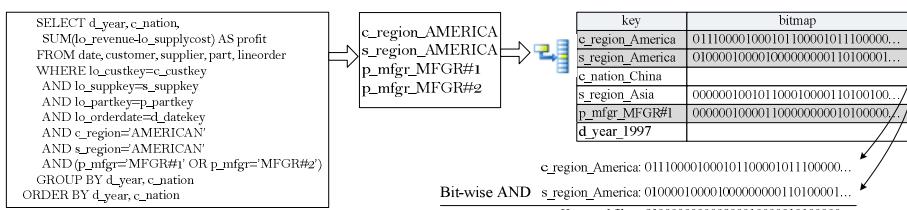


Fig. 1. Generate keyword filter

We transform traditional query tree in database into two level filtering. First of all, we divide query clauses into different processes:

- c_region='AMERICAN' AND s_region='AMERICAN' ↳ keyword filter
- p_mfgr='MFGR#1' OR p_mfgr='MFGR#2' ↳ predicate bitmap filter
- GROUP BY d_year ↳ group vector
- GROUP BY c_nation ↳ group vector

We generate predicate bitmap filter on *part* table for unmatched keywords 'p_mfgr_MFGR#1' and 'p_mfgr_MFGR#2' with the following SQL command:

```
SELECT CASE WHEN (p_category = 'MFGR#1' OR p_category = 'MFGR#2') 1 ELSE 0 FROM Part
```

For grouping attribute 'c_nation', we generate a group vector with same rows as *customer* table, in which 'c_nation' value with 'AMERICA' in 'c_region' attribute is recorded in correlated position of group vector. We use dictionary compression for group vector, store distinct values in dictionary and only record short value key in group vector. Grouping attribute 'd_year' has no correlated predicates on *date* table, so we directly project d_year attribute as group vector. Data warehouse commonly employs surrogate key[13] as primary key of dimension which is a sequential number used in TPC-H and SSB. Both predicate bitmap filter and group vectors have same rows as dimension table, the bitmap index or group vectors can be mapped to surrogate key. So foreign key in fact table can be directly mapped to correlated position in predicate bitmap filter and group vector[14,15].

Now we have one keyword bitmap filter for level-1 positional scan on fact table, one predicate bitmap filter for level-2 filtering to decide whether survived records satisfy the rest of predicates, result record directly extract grouping keys from two group vectors.

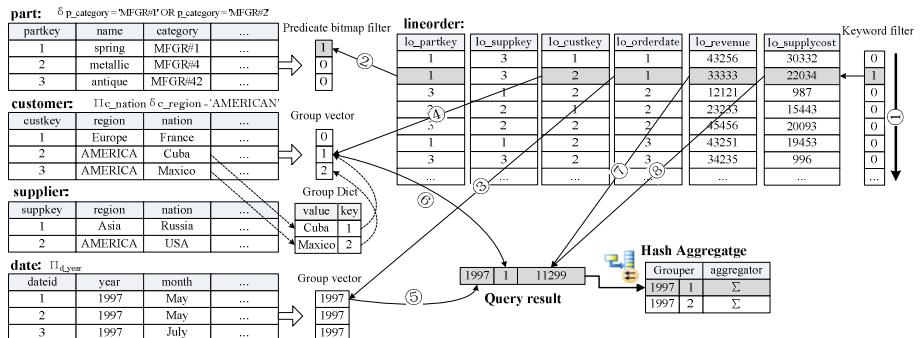


Fig. 2. Two level filtering for analytical processing

Figure 2 illustrates two level filtering for analytical processing. Step ① denotes positional scan on big fact table with generated keyword filter to skip records with '0' positions; in step ②, foreign key 1 in lo_partkey attribute locates 1st position in predicate bitmap filter; if the correlated bit is 1 then get lo_orderdate attribute value 1 to locate 1st item in group vector d_year in step ③ and get lo_custkey attribute value 2

to locate 2nd item in group vector c_nation in step ④; otherwise, skip current record and go to step ①; in step ⑤ and ⑥, correlated items 1997 and nation key 1 are extracted as hash key; in step ⑦ and ⑧, correlated measure values in lo_revenue and lo_suppcost are extracted and executed lo_revenue-lo_suppcost as result measure value. Finally, query result with hash keys and measure value is pushed into aggregate hash table for aggregation.

3.4 Multicore Parallel Processing

Our two level filter processing is based on sequential scan on bitmap and positional scan on fact table with predicate bitmap filters, and it is easy to be divided into parallel processing with horizontal partitions. The keyword bitmaps bit-wise combination process can also be converted to parallel processing with horizontal partitions. Figure 3 illustrates the multicore parallel processing for two level filtering.

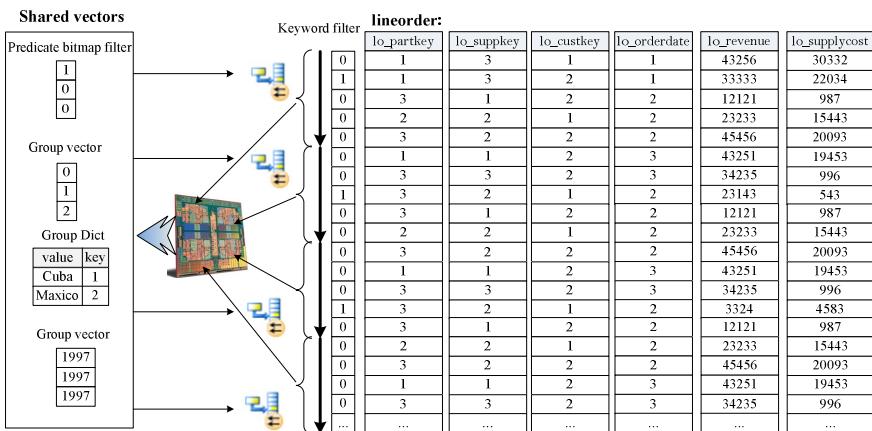


Fig. 3. Parallel two level filtering for analytical processing

We divided keyword filter into several partitions according to available cores of processors with offset addresses of bitmap. Each core scans specified bitmap partition, locates ‘1’ position on fact table, executes predicate bitmap filtering and aggregate operation in parallel. Finally, we start a global merging process to gather result sets and make final aggregate. In the parallel processing, predicate bitmap filters and group vectors are shared for all cores, each core maintains private aggregate hash table during processing isolated from other cores. After parallel processing, the global merging is efficient because the analytical results are very small aggregate datasets.

3.5 System Storage Model

In our prototype system, we divide tables into two kinds of storage models: dimension tables are much smaller than fact table but involve different data types and many complex predicates, we adopt pluggable database engine for dimension tables such as

PostgreSQL or MonetDB; fact table is very big with simple data type on foreign key columns and measure columns, we use an array based column store for efficiency on storage and access(positional access by array index). Hot keywords act as hot data index, which can dramatically reduce big fact table scan cost and can simplify complex query as keyword search style processing.

If RAM capacity is much smaller than SSD capacity, for example Oracle Exadata X3 Database In-Memory Machine has 256GB RAM and 1.6TB flash memory, we can use the relative small RAM as key(bitmap) store and use large flash memory as big fact table storage. Pre-generated predicate bitmap filters and group vectors are also RAM resident, so we can perform efficient keyword bitmap combination with bit-wise operators, high selective positional scan on flash memory, low latency predicate bitmap filtering and efficient aggregation in RAM. With storage strategy in figure 4, only processing related vectors and hot data index(keyword bitmaps) need to be RAM resident, big fact data with hot data areas are stored in lower level and large storage with efficient positional scan, dimension tables can be RAM resident or flash memory resident according to RAM capacity, we can get tradeoff between high performance and low cost.

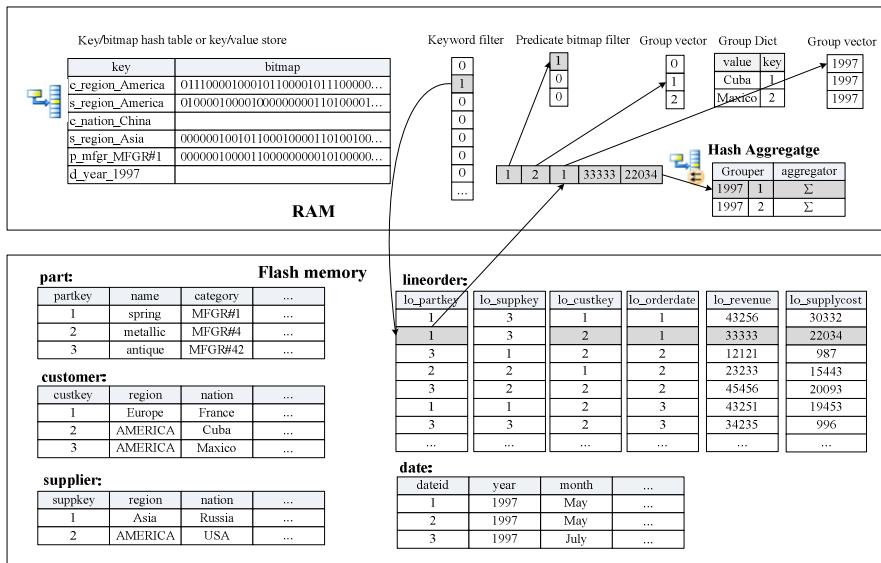


Fig. 4. RAM-Flash memory storage level

3.6 Update for Keyword Oriented Bitmap Join Index

We always maintain TOP K hot keywords in key(bitmap) store, if existed keywords are no longer in hot keyword candidates, the key(bitmap) can be swapped to lower level storage gradually to free RAM space and can be reused when it is accessed. When keywords are non-frequent, they can be removed permanently. When fact table is updated with fresh data(append-only mode), we can still use existed key(bitmap)

pairs with partial positional scan on indexed records and sequential scan on fresh records. When key(bitmap) pairs are to be updated, we can just use fresh fact data to append bitmaps on existed bitmaps.

4 Experiments

Our experiments are conducted on a HP ProLiant BL460c G7 server with two Intel® Xeon® Processor E5645 (6-core, 12M Cache, 2.40 GHz), 48GB memory and 300G 10K rpm 2.5' SAS hard disk. The OS version is ubuntu-11.10-server-amd64. We use star schema benchmark(SSB) dataset with SF=100 with standard SSB data generator, the row number of fact table is 600,000,000, and four dimension tables only takes up 0.84% in total size.

We store dimension tables in MonetDB for generating predicate bitmap filters and group vectors, we also use MonetDB as performance benchmark because MonetDB is an in-memory column-store analytical database which is well known for its excellent performance. We load fact table into array oriented column store to provide positional scan according to keyword bitmap filter. Keyword bitmap join index uses hash table for key(bitmap) pairs storage and access.

4.1 Overall Performance

We divide the whole processing into two large parts:

- (1) Bitmap combination. In this phase, we analyze SQL statement and extract keywords from **WHERE** clauses for keywords. We probe key(bitmap) store for bitmaps from keywords, and then execute a bit-wise AND for existed bitmap combination. We use keyword bitmaps from AND clauses, if some keyword bitmaps are missing in an OR clause, we neglect the existed keywords. In order to simplify system design and measure maximal benefit of keyword filtering, we create bitmaps for all keywords in SSB testing queries.
- (2) Processing. We use multicore parallel processing for maximal performance. Sequential scan on keyword filter is divided into parallel processing with logical partition from start offset to end offset. Each core scans the small bitmap partition first to skip non-matched positions in fact table, the survived records in fact table are then pushed to predicate bitmap filters for the second filtering, only records go through all the predicate bitmap filters are to be aggregated. In our TOP K keywords identification strategy, high selective keywords have more possibility to create bitmap join index, so keyword filter can make scan on big fact table more efficient.

Figure 5 shows the processing time of the two phases. We can see that bitmap combination cost is very close to each other. Multicore parallel bitmap combination is very efficient. Average time of processing one bitmap is about 143ms.

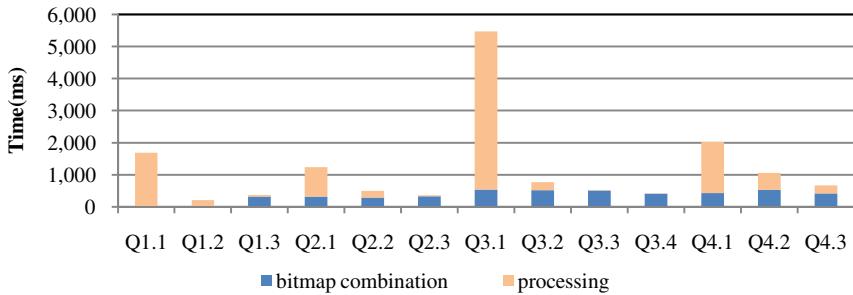


Fig. 5. Performance of different processing phases

Different queries have different complexity for the second level filtering and aggregation. Queries with higher selectivity still cost more time for predicate bitmap filtering and the following aggregation like Q2.1, Q3.1 and Q4.1. For other low selectivity queries, we can see that the first level keyword filtering drops much of processing cost and the second level filtering is very efficient. We see that keyword filtering for general high selective queries is very efficient, we can further reduce bitmap combination time for better performance by GPGPU or Coprocessor with more cores and higher parallel processing power.

We also measure performance without keyword bitmaps. If we have no available keyword bitmaps, the process is converted to scan foreign keys in fact table, pushing each record to predicate bitmap filters and select final record for aggregate. In this process, traditional hash join is simplified as bitmap filtering, it is simple and efficient.

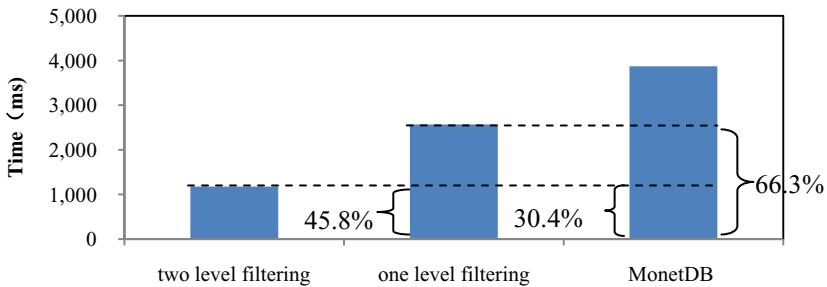


Fig. 6. Average SSB Query Time Comparison

We also use MonetDB as benchmark database with same dataset. Each query is continuously performed 5 times to select the minimal execution time as in-memory processing time to avoid disk I/O influence. Figure 6 shows the average processing time of 13 SSB queries for each workload. The keyword filter supported two level filtering costs average 30.4% query time as MonetDB, and costs 66.3% time of MonetDB without keyword bitmaps. Keyword bitmaps costs 45.8% time of one level filtering process.

4.2 Performance Analysis

Table 1 shows the detailed processing information. Bitmap combination is almost linear time, and we can predict bitmap combination time according to bitmap numbers. Column of “two level filtering” equals to column “bitmap combination” plus column “processing”, column “one level filtering” denotes processing without keyword bitmaps, column “selectivity” denotes total selectivity by all keywords, and column “Improvement ratio” is calculated by “(one level filtering- two level filtering)/ two level filtering” to measure the improvement for two level filtering compared with process without keyword bitmaps.

Bitmap combination time is 30.27% in total processing time. We can shorten this time by using less selective keywords instead of using all available keywords, we can also improve bit-wise operation performance by storing keyword bitmap in GPGPU or Coprocessor(e.g., Intel® Xeon Phi™ coprocessor 5110P) for even shorter time. If we compress bitmap with selectivity of 1/100, using SHORT INT(according to maximal offset value) vector to record non-zero positions, we can compress bitmap size to 16% of original size($1\% * N * 16\text{-bit}/N\text{-bit}$). With compression, we can store more keyword bitmaps in high performance coprocessor.

The minimal performance improvement is 18.68% in Q3.1 with selectivity 3.4%, and the maximal performance improvement is 646.85% in Q2.3 with selectivity of 0.02%. The general rule is that keyword filtering improves more with lower selectivity query. Average improvement for keyword filtering is 118.37%.

Table 1. Performance analysis

Queries	bitmap number	bitmap combination	processing	two level filtering	one level filtering	selectivity	Improvement ratio
Q1.1	1		1685.7	1685.7	3099.79	1.90%	83.89%
Q1.2	1		213.746	213.746	1485.19	0.07%	594.84%
Q1.3	2	316.98	52.018	368.998	1390.15	0.01%	276.74%
Q2.1	2	321.545	912.818	1234.363	3302.74	0.80%	167.57%
Q2.2	2	293.853	205.482	499.335	2729.3	0.16%	446.59%
Q2.3	2	326.543	29.889	356.432	2662.03	0.02%	646.85%
Q3.1	4	538.668	4924.49	5463.158	6483.56	3.40%	18.68%
Q3.2	4	524.566	244.246	768.812	1773.32	0.14%	130.66%
Q3.3	4	507.084	13.64	520.724	1469.69	0.01%	182.24%
Q3.4	3	412.779	1.393	414.172	1485.57	0.000076%	258.68%
Q4.1	3	435.397	1596.18	2031.577	3667.11	1.60%	80.51%
Q4.2	4	526.188	534.968	1061.156	2658.61	0.46%	150.54%
Q4.3	3	422.833	244.328	667.161	1171.01	0.01%	75.52%
Average Time		355.88	819.92	1175.79	2567.54		118.37%

5 Conclusions

In analytical workloads, skewed accesses form hot datasets defined by multiple keywords. Creating bitmap join index for these hot keywords can dramatically reduce join cost for queries with same keywords. We first set up keyword bitmap join index

mechanism to manage bitmap join index for global keywords to improve index storage efficiency; multidimensional queries are converted to keyword bitmap filtering and predicate bitmap filtering to improve query processing efficiency. Keyword bitmap filtering improves efficiency of big fact table scan, and can be also used as memory resident index while big fact table can be stored in large flash memory to reduce storage cost.

Acknowledgements. This work is supported by the Fundamental Research Funds for the Central Universities (the Research Funds of Renmin University of China, Grant No.12XNQ072 and project of “Key Research of Big Data In-memory Analytical Processing”), the National Natural Science Foundation of China (Grant No.61170013), Heilongjiang technical project of GC12A307, the Program for “New Century Excellent Talents” and the RUC-NEC joint project on In-memory Column Store Techniques.

References

1. <http://www.sap.com/solutions/technology/in-memory-computing-platform/hana/overview/index.epx>
2. <http://www.oracle.com/us/products/database/exadata-db-machine-x3-2-1851253.pdf>
3. Boncz, P.A., Mangegold, S., Kersten, M.L.: Database architecture optimized for the new bottleneck: Memory access. In: VLDB, pp. 266–277 (1999)
4. Funke, F., Kemper, A., Neumann, T.: HyPer-sonic Combined Transaction AND Query Processing. PVLDB 4(12), 1367–1370 (2011)
5. O’Neil, P., O’Neil, B., Chen, X.: The Star Schema Benchmark (SSB), <http://www.cs.umb.edu/~poneil/StarSchemaB.PDF>
6. http://docs.oracle.com/cd/B10500_01/server.920/a96520/indexes.htm
7. http://en.wikipedia.org/wiki/Power_law
8. Levandoski, J., Larson, P., Stoica, R.: Identifying Hot and Cold Data in Main-Memory Databases. In: ICDE 2013 (2013)
9. Park, D., Du, D.H.C.: Hot data identification for flash-based storage systems using multiple bloom filters. In: Proceedings of the 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies, MSST 2011, pp. 1–11 (2011)
10. Aouiche, K., Darmont, J., Boussaid, O., Bentayeb, F.: Automatic Selection of Bitmap Join Indexes in Data Warehouses. CoRR abs/cs/0703113 (2007)
11. <http://memcached.org/>
12. <http://redis.io/>
13. http://en.wikipedia.org/wiki/Surrogate_key
14. Zhang, Y., Wang, S., Lu, J.: Improving performance by creating a native join-index for OLAP. Frontiers of Computer Science in China 5(2), 236–249 (2011)
15. Abadi, D.J., Madden, S., Hachem, N.: Column-stores vs. row-stores: how different are they really? In: SIGMOD Conference 2008, pp. 967–980 (2008)

Can SSDs Help Reduce Random I/Os in Hash Joins?

Liang Huai Yang¹, Mingchao Liu¹, Yifan Pan¹, Weihua Gong¹, and Simon Stannus²

¹ School of Computer Science & Technology, Zhejiang Univ. of Technology, Hangzhou, China

² School of Computing and Information Systems, University of Tasmania, Tasmania, Australia

Abstract. A hybrid drive combines the features of SSDs and HDDs in the same unit by using SSD as the intermediate cache. In light of its promising features, we propose a new algorithm called CGHJ(Cached Grace Hash Join) for hybrid drives, which reduces hard disk random I/Os that occur in the partitioning phase of traditional Grace hash joins by caching the segments of buckets in the SSD and migrating these segments of each partition contiguously to the HDD. Experiment results show that CGHJ can greatly reduce random disk I/Os in the case of small joining working spaces or larger relations and improve hash join performance.

Keywords: Hash Join, query processing, SSD buffer, hybrid storage.

1 Introduction

Join is an expensive core operation in DBMSs and its implementation dictates the overall performance of a DBMS. Hash join is popular and shown to outperform sort-merge join in many situations. With the emergence of flash memory, two trends of flash usage in DBMS are currently observed, i.e., flash-only or hybrid flash-HDD systems. Flash-based DBMSs became a hot research topic. A concise review on effective use of SSDs in DBMSs was given in other works [1,7]. All works generally leverage the fast random reads of flash SSDs and avoid random writes.

Several flash-aware join algorithms for flash-only storage have been proposed. RARE-join [5] uses PAX page layout to reduce I/Os by only fetching the join columns and PAX mini-pages while taking advantage of SSDs' low random read latency. However, authors only address this idea theoretically. DigestJoin [1] is similar to RARE-join except that it uses the traditional row-oriented page layout. FlashJoin[6] is a pipelined multi-table join algorithm optimized for SSDs by using join indexes and late materialization to reduce memory usage and data overflow. Hash join performance with SSDs in commercial DBMSs was assessed in [4] and the superiority of SSDs was shown.

The most related work to ours is Seq+[3], which reduced random HDD I/Os in the hash join by using batch-write and write- and read-group techniques, and its effectiveness was proved by theoretical analysis. As SSDs fill the latency gap between RAM and HDDs, we believe that it is worthwhile to revisit the issue of reducing random I/Os by using an SSD as the cache.

In this paper, we propose a new algorithm called CGHJ (Cached Grace Hash Join) for hybrid drives, which reduces hard disk random I/Os that occur in the partitioning phase of the traditional GHJ by caching the segments of buckets in the SSD and then migrating all these segments of each partition contiguously to the HDD.

The rest of this paper is organized as follows: Section 2 details the CGHJ algorithm by using an SSD as the cache between the RAM and HDD. Experiments are conducted in Section 3 and we conclude in Section 4.

2 Cached Grace Hash Join

The outstanding I/O throughput gap between HDDs and SSDs makes SSDs ideal for caching data. Based on this, we propose a hash join algorithm called Cached Grace Hash Join (CGHJ for short) for hybrid drives, which reduces hard disk random I/Os in the partitioning phase of the traditional GHJ algorithm by caching the segments of buckets in an SSD and migrating these segments of each partition contiguously to the HDD later. In the partitioning phase, when a bucket buffer is full, instead of writing to the hard disk, CGHJ buffers the content to a temporary file on the SSD. Since random writes may cause unnecessary erase operations or even “write amplification” on SSDs, and thus degrade SSD performance, CGHJ adopts the append-only method to avoid random writes on the SSD. When the SSD buffer is full, the segments, which belong to each partition, will be migrated to the corresponding bucket file on the hard disk in chunks the size of the migration buffer. After completing the migration, the SSD buffer is emptied and the partitioning process continues until no tuples are left in relations. The partitioning process is shown in Fig. 1. Using this scheme, CGHJ uses the SSD as a cache to reduce time-consuming random accesses to the HDD and take advantage of the low access latency and high random read throughput of SSDs. The join phase of CGHJ remains the same as that in the GHJ algorithm.

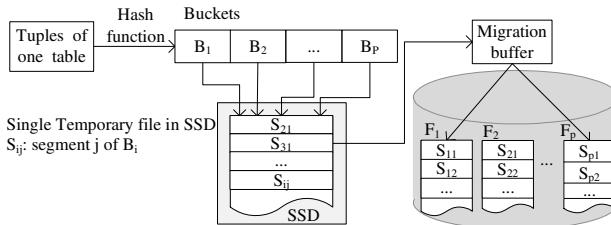


Fig. 1. The Partitioning Process of Cached Grace Hash Join

Here we borrow some I/O analysis results from [3]. Given two relations R and S to be joined, we assume that the smaller one R is the inner relation, and the other S is the outer relation. Let M be the total size of the memory buffer, M_{IB} , the input buffer size, M_B , the total bucket buffer size, and $|R|$ ($|S|$), the size of relation R (S). Let N be the number of partitions, so the buffer size for each bucket is M_B/N . The cost of one random block access is denoted as A_{ran} , that of one sequential block access, A_{seq} . The ratio A_{ran}/A_{seq} is denoted as ρ . Let SSD_{buffer} be the size of SSD buffer. CGHJ actually

increases the bucket buffer from M_B/N to SSD_{buffer}/N . It will not start to write to the HDD until the SSD buffer runs out of space. Therefore, the effect of such a scheme is tantamount to expanding the input buffer size from M_{IB} to SSD_{buffer} in partitioning phase. Hence, the I/O cost for reading R and S in the partitioning phase becomes:

$$C_{PR}^{CGHJ} = (|R| + |S|)A_{seq} \frac{SSD_{buffer} + \rho - 1}{SSD_{buffer}} \quad (1)$$

The cost for writing back all the buckets to the bucket files on the HDD is:

$$C_{PW}^{CGHJ} = (|R| + |S|)A_{seq} \frac{SSD_{buffer} + N(\rho - 1)}{SSD_{buffer}} \quad (2)$$

$C_{migration}^{CGHJ}$ denotes the cost of migration the cached bucket data from SSD to HDD. CGHJ is designed for the condition that there is limited working space and the SSD buffer is larger than memory while smaller than the inner relation. Given that SSD_{buffer} is much larger than M_B and M_{IB} , from Formulae 1, 2 and 3 we can compute the cost saved by CGHJ in partitioning phase as follows:

$$C_{save} = C_{PR} - C_{PR}^{CGHJ} + C_{PW} - C_{PW}^{CGHJ} - C_{migration}^{CGHJ} \quad (3)$$

After some calculations, it arrives at:

$$C_{save} = (N + 1)(|R| + |S|)(A_{ran} - A_{seq})\left(\frac{1}{M_{IB}} - \frac{1}{SSD_{buffer}}\right) - C_{migration}^{CGHJ} \quad (4)$$

Formula 4 implies that the cost saving of CGHJ comes from 3 factors: HDD characteristics, SSD buffer size, relation sizes. In terms of the characteristics of the HDD, the larger the cost difference between random access and sequential access ($A_{ran} - A_{seq}$) is, the faster CGHJ will be. However, this factor depends solely on the characteristics of the HDD. The cost saving also depends on the difference between the inverse of the input buffer size and the inverse of the SSD buffer size, i.e., $\left(\frac{1}{M_{IB}} - \frac{1}{SSD_{buffer}}\right)$. So a larger SSD buffer increases the cost saving. As to the relation size, larger relations will contribute more to the cost saving of CGHJ.

On the other hand, from Formula 4, we can infer that the sequential access cost of two relations in the HDD has been deducted from the final cost and the remainder is the I/O cost saved by CGHJ. CGHJ needs one extra pass of sequential write and random read on the SSD compared to GHJ; however, considering that the bandwidth of SSDs is much higher than that of HDDs, and that this paper assumes that the working space is still limited and thus will result in a large number of random I/Os, the extra I/O cost on the SSD will be counteracted by the saved cost of CGHJ through reducing the random I/Os on the HDD. This will be validated in the experiment part. Generally speaking, a larger SSD_{buffer} will lead to higher cost savings.

3 Experimental Evaluation

The GHJ and CGHJ algorithms were implemented in C++ with no special optimizations incorporated. Our experiments were conducted under the Windows 7 with an

Intel Core 2 Quad Core Q8200 2.33GHz processor, 4GB main memory, Seagate 7200RPM 500GB HDD, and two SSDs of different brands: a Samsung 830 Series 128GB SSD and an Apacer 32GB SSD. BIOS AHCI option is enabled to achieve highest performance of SSDs. Table 1 and Table 3 illustrate the I/O performance of our Seagate HDD as tested by the hard disk utility *HD Tune*. The I/O performance measurements of the SSD are referred to in Table 4, as measured by *AS SSD Benchmark 1.6*. CGHJ uses a migration buffer of size 256KB as the default except where specified otherwise.

In our experiments, the relations to be joined are generated by the data generator of the TPC-H benchmark with different scales from 1 to 30. Hash join is performed on ORDERS and CUSTOMER relations, with ORDERS as the outer relation and CUSTOMER as the inner relation. The join attributes of two relations are the primary key of CUSTOMER and the foreign key of ORDERS respectively. The sizes of CUSTOMER and ORDERS under 4 different scales are shown in Table 2.

Table 1. Random I/O performance of the HDD

I/O Block size (KB)	IOPS	Average throughput (MB/S)
0.5	56	0.027
4	56	0.222
64	54	3.388
1024	40	40.673

Table 2. Sizes of Two Relations and Partitions under Different Dataset Scales

	Data Scale			
	1	10	20	30
CUSTOMER(MB)	26	257	516	774
ORDERS(MB)	186	1344	3789	5691

Table 3. The I/O Characteristics of the HDD (measured by *HD Tune*)

Brand/model	Capacity	SR(MB/s)	SW(MB/s)	RR(IOPS)	RW(IOPS)
Seagate 7200 RPM HDD	500GB	101	101	56	56

Table 4. The I/O Characteristics of the SSDs (measured by *AS SSD Benchmark 1.6*)

Brand/model	Capacity	SR(MB/s)	SW(MB/s)	RR(MB/S)	RW(MB/S)
Samsung 830 SSD	128GB	255	235	168	68
Apacer SSD	32GB	140	89	15	1.5

SR=Sequential Read; SW= Sequential Write; RR=Random Read; RW=Random Write

3.1 Effects of Dataset Scales on CGHJ

The first set of experiments examined the impact of 4 dataset scales on CGHJ and the conventional GHJ algorithm by fixing the memory buffer size at 3MB under two different configurations of the hybrid storage system: Apacer SSD + HDD and Samsung SSD + HDD. The effects of various migration buffer sizes on performance will be evaluated in another section.

From Table 1, we find that the random read/write performance of the HDD is acceptable when the data I/O block size is between 64KB to 1024KB. So to be realistic, each partition cached in the SSD should be at least multiples of this block size. Also, to be fair in comparing the performance of CGHJ and GHJ among different dataset

scales, we decide to allocate the SSD buffer in size proportional to the dataset scale, i.e., the SSD buffer size is allocated to be $N \times 512\text{KB}$, where the number of partitions is N . The experiment results are shown in Fig. 2.

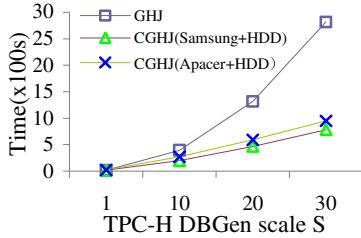


Fig. 2. Performance Comparisons on Datasets of Different Scales

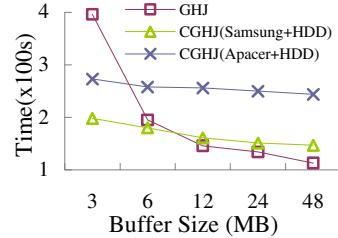


Fig. 3. Performance under Various Buffer Sizes with Dataset Scale=10

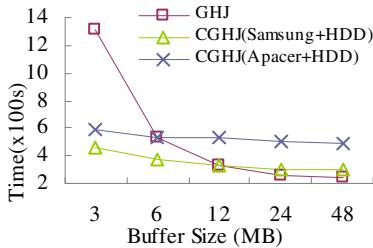


Fig. 4. Performance under Various Buffer Sizes with Dataset Scale=20

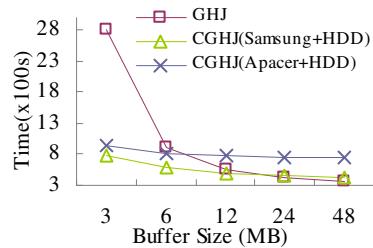


Fig. 5. Performance under Various Buffer Sizes with Dataset Scale=30

The Y-axis shows the total join time of CGHJ and GHJ. The result shows that the larger the input relation is, the higher cost saving CGHJ can achieve. When the memory buffer size is fixed, a larger relation results in more random I/Os in the partitioning phase for the GHJ algorithm and creates more opportunities for CGHJ to reduce random I/Os. Also, it can be seen that the performance for CGHJ on the two different configurations is different. Due to the higher random read/write throughput of the Samsung SSD, CGHJ on the Samsung SSD outperforms that on the Apacer SSD. For the Samsung SSD + HDD, CGHJ is 1 to 4 times faster than GHJ; for the Apacer SSD + HDD, CGHJ is 1.5 to 3 times faster than GHJ for scales 10, 20, and 30, with an exception at scale 1. For the dataset at scale 1, the relations are very small and CGHJ will not benefit from the low-end SSD + HDD combination. With this memory buffer size, each bucket buffer ($3\text{MB}/11 \approx 279\text{KB}$) approaches an ideal I/O block size for random writing on the HDD and therefore there is not so much room left for CGHJ to reduce random I/Os. Moreover, since the dataset at scale 1 can be wholly held in the SSD there is no need to handle it this way, so we did not evaluate the experiments on this dataset any further.

3.2 Effects of Various Memory Sizes on CGHJ

The experiments of this section explore the effects of various memory sizes on the performance of CGHJ. The results of scales 10, 20, and 30 are shown in Fig. 3, Fig. 4, and Fig. 5 respectively. It can be seen that the performances of all algorithms improve with the increase in buffer size. However, when the buffer increases to a certain limit, the performance of CGHJ reaches a plateau. The reason is that as the size of the buffer increases, the size of each bucket's buffer increases as well, and this will decrease the number of random I/Os in the partitioning phase.

4 Conclusions

This paper explored the possibility of using an SSD to reduce the random I/Os in hash joins as a part of a hybrid storage system. Our proposed scheme *CGHJ* reduces random I/Os on the HDD by using the SSD as cache between the RAM and HDD. CGHJ works well in situations where GHJ would face a large number of random I/Os due to limited memory or relations are large. And this idea could be extended to hybrid hash joins as well.

Acknowledgements. This work was supported by National Natural Science Foundation of China (Grant No. 61070042) and Zhejiang Provincial Natural Science Foundation (Grant No. Y1090096 and Y13F020114).

References

1. Athanassoulis, M., Ailamaki, A., Chen, S., Gibbons, P.B., Stoica, R.: Flash in a DBMS: where and how? *IEEE Data Eng. Bull.* 33(4), 28–34 (2011)
2. Li, Y., Xu, J.L., Choi, B.: DigestJoin: exploiting fast random reads for flash-based joins. In: Proc. of Int'l Conf. on Mobile Data Management, pp. 152–161. IEEE Press, Washington (2009)
3. Lo, M.-L., Ravishankar, C.V.: Towards eliminating random I/O in hash joins. In: Proc. of Int'l Conf. of Data Eng., pp. 422–429. IEEE Press, Washington (1996)
4. Park, S.S., Lee, S.W.: Hash join in commercial database with flash memory SSD. In: Proc. of Int'l Conf. on Computer Science and Info. Technology, pp. 265–268. IEEE Press, Washington (2010)
5. Shah, M., Harizopoulos, S., Wiener, J., Graefe, G.: Fast scans and joins using flash drives. In: Proc. of DaMoN Conf., pp. 17–24. ACM Press, New York (2008)
6. Tsirgiannis, D., Harizopoulos, S., Shah, M.A., Wiener, J.L., Graefe, G.: Query processing techniques for solid state drives. In: Proc. of ACM SIGMOD Conf., pp. 59–72. ACM Press, New York (2009)
7. Koltsidas, I., Viglas, S.: Data management over flash memory. In: Proc. of SIGMOD Conf., pp. 1209–1212. ACM Press, New York (2011)

An Index Model for Multitenant Data Storage in SaaS

Pang Cheng, Li Qingzhong^{*}, and Kong Lanju

School of Computer Science and Technology

Shandong University

Jinan, China

pangchengsdu@yahoo.cn, {lqz, k1j}@sdu.edu.cn

Abstract. For SaaS applications which multiple tenants share sparse tables, fields of varying data types are stored in one flex column based on customization. Thus it is not practical to create native physical indices for tenants. In this paper, we put forward an indexing model for sparse table storage model based on pivot tables in multi-node environments. The core idea is copying field data to be indexed to another table and creating physical indices in that table. This model supports customization and isolation characteristics of multitenant applications, and provides load balance means in index subsystem, and controls the number of index data tables thus memory consuming reasonably.

1 Introduction

SaaS (Software as a Service) is a software delivery model in which software and associated data is centrally hosted on the cloud [5]. It is beneficial to leverage economy of scale to reduce average cost of ownership relative to on-premises solutions [4]. Among storage models for SaaS applications, sparse table model is widely used in which data of multiple tenants is stored in a shared sparse table and fields of varying data types can be stored in a flex column based on tenants' customization [3, 8, 9]. For a SaaS application, it is necessary to establish adequate index structures as data volumes increase steadily. However, most of existing indexing models are not designed for SaaS applications and cannot be aware of multitenant properties.

In this paper, we put forward a feasible multitenant indexing model through establishing appropriate shared and isolated storage for index data. We detach index data based on logic tables rather than tenants, thus preventing the skyrocketing increase of number of physical tables caused by the linear relativity between number of index data tables and number of tenants. And the index data tables are dense, consequently, the model is memory-saving, especially when multiple replicas exist [3]. In addition, we set a threshold for the number of indices that an index data table could sustain, thus achieving load balance and mitigating the negative effects of skew index types.

The rest paper is organized as follows. The related work is analyzed in section 2. In section 3, we introduce the system architecture and storage model. In section 4, we explain isolation and maintenance strategies for the model. Section 5 shows experimental analyses. Section 6 concludes the whole paper and explores future work.

^{*} Corresponding author.

2 Related Work

Chen [10] proposed a storage model based on multiple sparse tables; each table possesses various numbers of columns by estimates. The most suitable one is chosen to reduce nulls. Obviously, the model cannot dynamically adapt to customization.

In Force.com platform [1], indices are implemented by synchronously copying data marked for indexing to the corresponding index data table which is based on a pivot table. However, the paper does not describe the schema of index data tables and the isolation strategy of index data. It supplies a direction for a multitenant index model.

Mei Hui [2] provides Multi-Separated Indexing; they build an index for each tenant rather than for all tenants. It is not efficient since the number of indices grows linearly with the number of tenants and too many indices are created with fierce competitions.

The Cloud Global Index [6] establishes local indexes for each data node and publishes some index nodes to the overlay network; thus local indices are located through searches on global indices. However, index data should resident in main memory; and redundant queries exist owing to broadcasting query patterns.

3 System Architecture and Storage Model

Our underlying storage model is based on multiple data nodes, and sparse tables are created in each node on demand. In order to reduce number of distributed transactions, one replica of a tenant is stored and only stored in one data node; a data node can accommodate multiple replicas though. In a data node, data of a logical table which belongs to multiple tenants is stored in one sparse table. Mixing storage of business data inevitably increases the data volumes an operation manages. For a tenant, data of different logical tables is stored in different sparse tables in a data node.

Business data is stored in specific sparse table columns according to customization. Columns of a sparse table are of string types. We set an appropriate replica factor for business data and disperse these replicas to different data nodes to balance reading loads. The model supports creating replicas on tenant and logical table level.

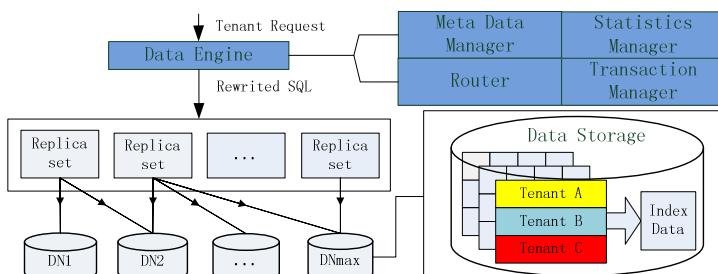


Fig. 1. System architecture

The system architecture is shown in Fig.1. The statistics manager collects requests and load statistics of data nodes to help to generate best execution plans. The transaction manager guarantees synchronization of replicas. The router transmits tenants' requests to appropriate data nodes to achieve load balance. In summary, the data engine receives requests, generates optimal execution plans based on metadata and statistics, and then fetches business data from appropriate data nodes; in the meantime, each data node should send table and index statistics to the statistics manager [11].

The data storage subsystem from the view of one data node is shown in Fig.2. The logical_ prefix indicates that the table has not been customized by tenants. The guid (globally unique identifier) column is the primary key of a sparse table; the tenID column marks the tenant a record belongs to, other flex columns record business data. Metadata within the dashed box locates in the data engine node.

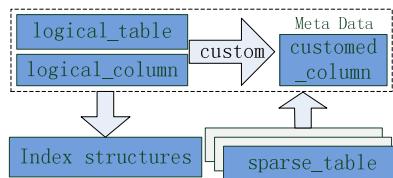


Fig. 2. Data storage in one data node

4 Index Model

4.1 Isolation Strategy

As one replica of a tenant is stored and only stored in one data node, we manage indices in each data node separately. It is unfeasible to create physical indices directly in sparse tables; therefore, we copy data marked for indexing to a corresponding index data table. Tenants can create indices independently of one another for a logical table. We will consider both isolated and shared storage mechanism for index data.

We notice that in SaaS applications, the number of logical tables is relatively stable (about dozens) and is smaller compared to that of tenants; and tenants tend to create indices of the same type (includes data types of the columns and unique constraints) in a logical table. In summary, isolating index data based on logical tables and index types but tenants could decrease the number of index data tables reasonably.

An index-related query is converted into two queries, one in the index data table and one in the sparse table. A physical index may be shared among multiple tenants, thus memory buffers for index pages are utilized efficiently, because buffering top index pages of a B+ Tree structure in memory will reduce the number of disk I/Os. As for insert or delete operations, owing to synchronization cost of source data, performances are degraded relative to those of operations with native physical indices.

4.2 Physical Structures

We create logical indices for each replica separately, and index data is stored in the same data node as the source data. An index structure from the view of a data node is shown in Fig.3. The metadata within the dashed box is stored in the data engine node.

Index data tables are created on demand. We can get the index data table corresponds to an index in the metadata table of index data table - avlb_idx_dat_tab. The degree column records the number of indices sustained by current table. Take the degree value and the record number of this table into account, the system creates a new index data table for current logical table and index type if necessary, thus balancing loads among all workable index data tables. The tenant_index table records metadata of all indices. We create a physical index on the idxID column and index data columns of each index data table to implement B+ Tree structure.

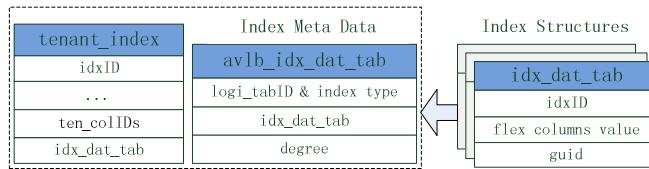


Fig. 3. Index structure in one data node

4.3 Index Management and Index Related DML

Index management operations include creating, dropping and rebuilding logical indices. Because an operation consists of a metadata management operation and a physical index management operation, it is no longer a pure data definition operation; business data and index data locates in different nodes to metadata; multiple replicas should also be considered; the ACID properties should be guaranteed by the data engine by consolidating corresponding physical operations to an index management operation into a distributed transaction. As creating an index, the most suitable index data table is selected according to degree values and record numbers of all available index data tables. As dropping an index, merge small index data tables if necessary.

Index related DML (Data Manipulation Language) operations include index-related query operations and synchronization operations between source data and index data brought on by inserting, updating or deleting source data of indexed columns. We modified the cost-based optimizer of Mysql to make it be aware of logic indices. As a replica and corresponding index data tables locate in the same data node, ACID properties of the database are guaranteed by consolidating all related physical operations of a logical operation into one local transaction. For each request, the data engine first rewrite the original SQL statement, namely, transform logical tables and columns into physical ones based on customization [7]. In query processing, the optimizer decides whether or not to adopt an index and select an optimal one based on real-time loads when alternate indices exist. If the adopted index is not a covering index, an index related query operation is transformed into two physical operations.

5 Experiments

In this section, we analyze our multitenant indexing model which is based on the data management platform of SaaS application delivery platform we have developed. This delivery platform supports ISVs to develop and deliver SaaS applications effectively in traditional development environments. All experiments are based on one replica of a logical table which locates in one data node entirely, as mentioned in section 4.1. We test average execution time of insert and query operations in the prototype system to verify validity of the index isolation strategy. Servers possess Intel X5620@2.4GHz processor, 8GB of memory, 500GB*2 of disk capacity with raid1, the operating systems are RHEL5, and the database is MySQL5.5. Clients possess Intel E7500@2.93GHz processor, 4GB of memory, 320GB of disk capacity.

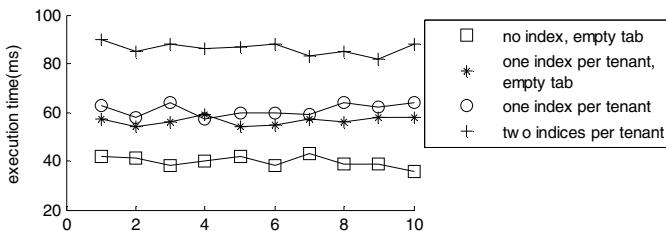


Fig. 4. Results of insert operations in multiple conditions

Each insert operation relates to one of 500 tenants. Each experiment is executed 20 rounds and each one consists of 100 insert operations; we record average execution time for each round, including time of rewriting a SQL statement. The result is shown in Fig.5. For Curve 1, the sparse table is empty and without indices. For Curve 2, the sparse table is empty and each tenant possesses a unique index which is built on a shared logical column. Contrasting these two curves, the performance decreases significantly owing to maintenance costs of index structures. The condition of curve 3 is same to that of curve 2 other than that each tenant possesses 5000 random records. Contrasting curve 3 with curve 2, the little differences indicate that the index structures can support large data volumes. The condition of curve 4 is same to that of curve 3 other than that a unique concatenated index is created for each tenant on another two shared logical columns. Contrasting curve 1, 3, 4, the execution time will increase correspondingly as number of indices increases.

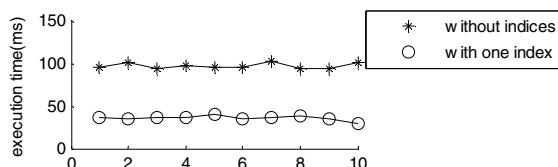


Fig. 5. Results of query operations in multiple conditions

Each query operation relates to one of 500 tenants, and each tenant possesses 5000 random records. Each experiment is executed 20 rounds and each one consists of 100 query operations. We record the average execution time of each round under condition of being without indices and being with a unique index for each tenant on a shared logical column. The cache is cleaned before each round. The result is shown in Fig.6. Apparently, indices significantly improve performance of query operations, indicating that the indexing model we proposed is effective in common use.

6 Conclusions

In this paper, we proposed a multitenant indexing model based on pivot tables, and the index data is isolated based on logical tables rather than tenants. The theoretical and experimental analyses show that this model improves performance of query operations significantly and performance losses of other operations are mild; and the number of index data tables is controlled reasonably. In the future, we will consummate hierarchies of the indexing model, tune the candidate index selection mechanism to further optimize performances of this model.

Acknowledgments. This work is funded by National Natural Science Foundation of China under Grant No. 61272241;Natural Science Foundation of Shandong Province of China under Grant No. ZR2010FQ010, No. ZR2010FQ026;Science and Technology Development Plan Project of Shandong Province No.2012GGX27036; Independent Innovation Foundation of Shandong University under Grant No. 2012TS075,No.2012TS074.

References

1. Weissman, C.D., Bobrowski, S.: The Design of the Force.com Multitenant Internet Application Development Platform. In: SIGMOD (2009)
2. Hui, M., Jiang, D., et al.: Supporting Database Applications as a Service. In: ICDE (2009)
3. Aulbach, S., Grust, T., Jacobs, D., Kemper, A., Rittinger, J.: Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques. In: SIGMOD (2008)
4. Aulbach, S., et al.: A Comparison of Flexible Schemas for SaaS. In: SIGMOD (2009)
5. wikipedia. Software as a service,
http://en.wikipedia.org/wiki/Software_as_a_service
6. Wu, S., Jiang, D., Ooi, B.C., Wu, K.-L.: Efficient B-tree Based Indexing for Cloud Data Processing. PVLDB 3(1), 1207–1218 (2010)
7. Li, J., Zhang, S., Liu, Z., Kong, L.: A Data Rights Control Model for a SaaS Application Delivery Platform. In: Mao, E., Xu, L., Tian, W. (eds.) ECICE 2012. AISC, vol. 146, pp. 139–146. Springer, Heidelberg (2012)
8. Beckmann, J.L., Halverson, A., Krishnamurthy, R., Naughton, J.F.: Extending RDBMSs To Support Sparse Datasets Using An Interpreted Attribute Storage Format. In: ICDE (2006)
9. Chu, E., Beckmann, J., Naughton, J.: The Case for a Wide-Table Approach to Manage Sparse Relational Data Sets. In: SIGMOD (2007)
10. Chen, W., Zhang, S., Kong, L.: A Multiple Sparse Tables Approach for Multitenant Data Storage in SaaS. In: IIS (2010)
11. Lanju, K., et al.: A research on SaaS-oriented multitenant indexing model based on key-value pairs. Chinese Journal of Computers (2010)

Reverse Top- k Group Nearest Neighbor Search

Tao Jiang¹, Yunjun Gao², Bin Zhang¹, Qing Liu², and Lu Chen²

¹ College of Mathematics Physics and Information Engineering, Jiaxing University,
Jiaxing 314000, China

{jiangtao_guido, zhangbin_selina}@yahoo.com.cn

² College of Computer Science, Zhejiang University, Hangzhou 310027, China
{gaoyj, liuqing1988, chenl}@zju.edu.cn

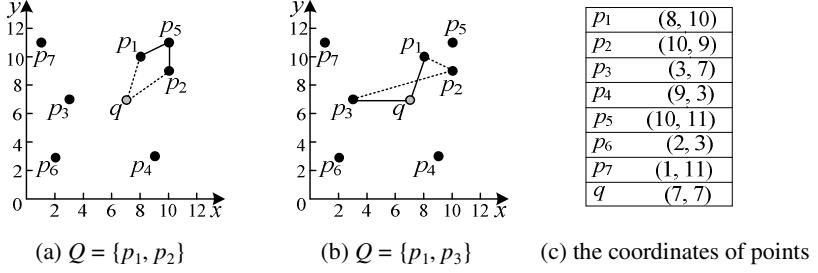
Abstract. This paper identifies and solves a novel query, namely, *Reverse Top- k Group Nearest Neighbor* (RkGNN) query. Given a data set P , a query object q , and two (user specified) parameters m and k , an RkGNN query finds k subsets, which have the least *aggregate distances*, such that each subset contains m data objects from P and has q in its *group nearest neighbor*. We formalize the RkGNN query. Then, we propose several algorithms for efficiently processing RkGNN queries. Our methods employ some effective *pruning heuristics* to prune away unqualified candidate subsets, utilize the *sorting* and *threshold mechanisms* to shrink the search space, and make use of the advantages of *lazy* and *spatial pruning techniques*. Extensive experiments with both real and synthetic datasets demonstrate the performance of the proposed algorithms in terms of effectiveness and efficiency.

1 Introduction

Given a set of data objects $P = \{p_1, p_2, \dots, p_n\}$ and a set of query objects $Q = \{q_1, q_2, \dots, q_m\}$, an *group nearest neighbor* (GNN) query^[1-8] returns a data object in dataset P with the smallest *sum* of distances to all data objects in Q . Figure 1(a) shows an example of group nearest neighbor over a dataset $P = \{p_1, p_2, \dots, p_7, q\}$ in a 2D space, where the coordinates (x-axis and y-axis) of each object in P are shown in Figure 1(c) and $Q = \{p_1, p_2\}$. Since the object p_5 has the minimum *aggregate distance* to Q , that is, $\text{adist}(p_5, Q)$, than other objects $p' \in P \setminus Q$ (i.e., the object q), it constitutes the GNN of Q on dataset P .

However, GNN query only finish the computation from the user's perspective. In this paper, we study *reverse group nearest neighbor* (RGNN) query which consider the GNN query from company's perspective. Given a dataset P and a query object q , an RGNN query aims to find multiple subsets from P such that each subset G has q as its GNN, and $|G|=m$ ($m \geq 2$), where m is a specified integer. When $m=1$, an RGNN query is reduced to an RNN query. In Figure 1(b), assume that $Q = \{p_1, p_3\}$ is the reference objects. Since $\{p_1, p_3\}$ has q as its GNN, it is a result of RGNN of q .

Unlike GNN query which only returns a data object as the answer, RGNN query might retrieve multiple subsets as the answers. To enhance user decision-making, we may set up a parameter k to limit the number of the subsets using the aggregate distance as the monotonic sorting function. This is a general RGNN query. We call it *reverse top- k group nearest neighbor* (RkGNN) query. Consider the example in Figure 1(b), RkGNN will return $\{p_1, p_3\}$ and $\{p_2, p_3\}$ as the answers when $k=2$.

**Fig. 1.** Illustration of group and reverse nearest neighbor

RkGNN query has many applications in business analytic and decision support systems. Consider the example: a big company plans to open m branches according to the location of headquarters (denotes as the query object q). The candidate locations of branches can be regarded as data objects in a database. Generally, the manager hopes that the branches have q as their GNN. In other words, these branches have less *sum* distances to q , and there is not another branch which has a smaller sum distances to the branches than q . Another example is: when designing a series of products (i.e., m products) from n alternative products picked up according to a specified prototype q , the corporation generally hopes that the sum distance from q is less for the group of products; meanwhile the new products should have a diversified design so that they can satisfy the requirement of different customer. The *diversity* can be measured by the sum distance from another product. Actually, the principle of design embodies the proximity with q and diversity from another object for a group of objects.

In this paper, we introduce two efficient algorithms, i.e., *Lazy RkGNN* (LRkGNN) and *Spatial pruning LRkGNN* (SLRkGNN) on datasets indexed by R-tree using Euclidean distance function, by several pruning heuristics, which reduce a large of subsets, the early stopping technologies which avoid searching the whole data space and the method of lazily outputting candidate subsets based sorting mechanism. Specifically, our main contributions are summarized as follows.

- (i) We formalize the RkGNN query, an interesting variant of GNN query.
- (ii) We develop two algorithms, viz. LRkGNN and SLRkGNN, for efficiently answering reverse top- k group nearest neighbor queries.
- (iii) We demonstrate the effectiveness of our proposed heuristics and the performance of our proposed algorithms by extensive experiments.

2 Problem Formulation

In this section, we formally define the reverse top- k group nearest neighbor query. Table 1 contains the primary symbols used in our description.

Definition 1 (Reverse top- k Group Nearest Neighbor, RkGNN). Given a dataset P , a query object q , and two (user specified) parameters m and k , an RkGNN query finds k subsets, which have the least *aggregate distances*, such that each subset contains m data objects from P and has q as its *group nearest neighbor*.

Table 1. Frequently used symbols

Notation	Description
m	the number of data objects in a subset
$\text{maxdist}(e)$	the maximum distance between lower left and upper right corner of $\text{MBR}(e)$
G	a subset contained in dataset P which contains m entries
G_{rlt}	a set of the results of RGNN query
G_{rfn}	a set of candidate combinations of RGNN query
best_kdist	the k -th aggregate distance of subsets in G_{rlt} in ascending order
best_hdist	the minimum aggregate distance of orderly outputting subsets produced by the objects in auxiliary heap H
best_rfndist	the minimum <i>aggregate distance</i> of all subsets in G_{rfn}

A straightforward approach is to enumerate the combination (or say subset G) of the data objects, and compute the GNNs of each such combination. However, this approach is not efficient.

3 RkGNN Query Processing

3.1 Lazy RkGNN Algorithm

In this subsection, we describe the *Lazy Reverse top- k Group Nearest Neighbor algorithm* (LRkGNN), which makes use of the *sorting* and *threshold mechanisms* to shrink the search space based *lazy techniques*, and employs some effective *pruning heuristics* to discard candidate subsets.

Lazy Output and Advanced Sorting Technique

Generally, basic RkGNN algorithm searches in a *best-first* manner [1], and uses the minimum distance between current entry e and the query object q as the sorting *key*. Moreover, it enumerates all subsets (or say combinations) using e and the entries among H when an entry e is removed from the auxiliary heap H . If the candidate subset G only contains data objects, it is processed by invoking MBM algorithm in [1] to judge whether q is the GNN of G ; otherwise G is inserted into the refined set G_{rfn} so that it will be processed later until the intermediate entry in G is expanded.

Unlike basic RkGNN algorithm, LRkGNN algorithm does not immediately enumerate the combinations for each time popped object until there are multiple data objects. LRkGNN uses two min-heaps, H_t and H_a , to save the data objects popped from

H satisfying (1), and all entries among H before enumerating, respectively. Then, LR k GNN generates all combinations so far using the data objects or entries of H_a .

Owing to that LR k GNN only need to retrieve k final results, LR k GNN uses the following inequality in (1) as the triggered conditions of enumerated combinations:

$$C_{|H_t|}^m \leq k \leq C_{|H_t|+1}^m, |H_t| \geq m+1 \quad (1),$$

where $|H_t|$ represents the number of data objects in H_t . The inequality in (1) guarantees that the algorithm generates at least k combinations; meanwhile it avoids to expand too many intermediate entries.

In fact, there still are too many entries in auxiliary heap H because a leaf node often includes more than one hundred data objects. Thus, the number of combinations is still very large, i.e., the number will be 161700 when $|H|=100$ and $m = 3$. To reduce the number of evaluating candidate subsets, LR k GNN progressively outputs some subsets from total subsets and process them according to their aggregate distances.

How to progressively generate some ordered subsets? Clearly, this problem is not trivial when the size of H_t is large. Our main idea is repeatedly applying two operations: decomposing and re-sorting. First, we sort the n entries in ascending order according their *key*, i.e., $\text{mindist}(e, q)$. Then, we decompose the set (i.e., C_n^m) into two parts (i.e., C_{n-1}^m and C_{n-1}^{m-1}) through a recursion formula in (2):

$$C_n^m = C_{n-1}^m + C_{n-1}^{m-1}, n > m. \quad (2).$$

Next, the two parts of subsets are merged and form a sorting list of subsets in ascending order according to $\text{adist}(q, G)$. Repeating the procedure until the aggregate distance of current subset is larger than the k -th aggregate distance of query results, namely, best_kdist . Our experiments show that only about 0.1% subsets need to be processed for each enumerating.

Example 1. Choose 2 objects from 4 ordered objects, o_1 , o_2 , o_3 , and o_4 , and their distances from q are 1, 2, 3, and 3.5, respectively. LR k GNN decomposes set C_4^2 into two sets, C_3^2 and $o_4 \cup C_3^1$. The former can be further decomposed two sets, C_2^2 (namely, $\{o_2, o_1\}$) and $o_3 \cup C_2^1$ (namely, $o_3 \cup o_2 \cup o_1$). The latter is $o_4 \cup o_3 \cup o_2 \cup o_1$, namely, $\{o_4, o_1\}$, $\{o_4, o_2\}$, and $\{o_4, o_3\}$. The set $\{o_2, o_1\}$ is firstly output, and then $\{o_3, o_1\}$ and $\{o_4, o_1\}$ is output. Finally, $\{o_3, o_2\}$, $\{o_4, o_2\}$, and $\{o_4, o_3\}$ are outputted as results.

Sorting and Threshold Method

Obviously, two key factors ultimately determine the actual performance of LR k GNN: how to progressively sort for all subsets formed by the entries among H , which might severely influence the number of subsets to be evaluated, and the strategy of choosing the stop point, which greatly shrink the search space.

Now, let's consider the second factor now. By Example 1, we know that LR k GNN progressively processes three parts of ordered subsets. Therefore, we use the variable best_hdist to save the minimum aggregate distance of each part of subsets. Moreover, when a new part of subsets are produced, best_hdist is also updated. For instance,

best_hdist is 3 when LRkGNN deals with the first part of subsets in Example 1. However, *best_hdist* becomes 4 when LRkGNN processed the second part of subsets.

In fact, LRkGNN can terminate the search using the following Theorem 1 and Corollary 1.

Theorem 1. RkGNN can stop the search if *best_hdist* and *best_rfndist* are both larger than *best_kdist*, namely, $\text{best_hdist} \geq \text{best_kdist}$ and $\text{best_rfndist} \geq \text{best_kdist}$.

Proof. Since *best_rfndist* is larger than *best_kdist*, there is not any combination which has a lower aggregate distance than *best_kdist* so far. Therefore, any combination among G_{rfn} is not the result of RkGNN query. On the other hand, since *best_hdist* $\geq \text{best_kdist}$, there is not any new combination which has a less aggregate distance than *best_kdist*. So, the Theorem 1 is correct based on above two cases. \square

Corollary 1. The search of RkGNN can be stopped if *best_hdist* is larger than *best_kdist*, namely, $\text{best_hdist} \geq \text{best_kdist}$, when the set of G_{rfn} is empty.

In practice, LRkGNN can also stop enumerating procedure by following Lemma 1.

Lemma 1. RkGNN can ignore the rest of processing subsets if the current subset G being processed has a larger aggregate distance than *best_kdist*, namely, $\text{adist}(q, G) \geq \text{best_kdist}$.

Proof. Since all enumerated subsets are arranged in ascending order according to their aggregate distance, the subsets after the current subset G have a larger aggregate distance. Thus, there is not another subset which has a larger aggregate distance than that of G if $\text{adist}(q, G) \geq \text{best_kdist}$. Therefore, the Lemma 1 is correct. \square

Theorem 1, Corollary 1 and Lemma 1 is not trivial because they guarantee that the algorithm save much unnecessary cost of computation.

Pruning Heuristics

Next, we introduce some pruning heuristics to discard many candidate subsets by Lemma 2 and Lemma 3.

Lemma 2. Assume that a node e contains at least $m+1$ data objects, and $\text{mindist}(q, e)$ denotes the minimum distance of node e from query object q . Any subsets among e cannot be the result of RkGNN query if $\text{maxdist}(e) \leq \text{mindist}(q, e)$.

Proof. Since the minimum distance of e from q is $\text{mindist}(q, e)$, we have $\text{mindist}(q, e)*m \leq \text{adist}(q, G)$ where all objects in G are from e . Assume that there is another point p' among e which is not contained in G . Obviously, the sum of distances between p' and data objects among G , $\text{adist}(p', G)$ is less than or equal to $\text{maxdist}(e)*m$. Thus, $\text{adist}(p', G) \leq \text{maxdist}(e)*m \leq \text{mindist}(q, e)*m \leq \text{adist}(q, G)$. Simplifying the inequality, we have $\text{adist}(p', G) \leq \text{adist}(q, G)$. This indicates that q is not the GNN of G . Since the subset G is a general assumption, therefore, we have that Lemma 2 is correct. \square

Consider the example in Fig. 2(a) where the minimum bounding rectangle (MBR) of node e contains three data objects, p_1 , p_2 , and p_5 . If the parameter m is 2, the node e will produce three combinations, $\{p_1, p_2\}$, $\{p_1, p_5\}$, and $\{p_2, p_5\}$. Because of $\text{maxdist}(e) \leq \text{mindist}(q, e)$, the three subsets of e , are not the result of RkGNN query. Although Lemma 2 can prune many combinations, it might be too restrictive to prune any combination when the MBR of the node is too large. At this point, we can still utilize the following Lemma 3 to prune the current subset G .

Lemma 3. If $\text{maxdist}(e)^*m$ is less than or equal to $\text{adist}(q, G)$, then the current subset G cannot be the result of RkGNN query.

Proof. Assume that there is another point p' among e which is not contained in G . Since $\text{maxdist}(e)$ represents the maximum distance between any two data objects among e , the sum of distances between p' and data objects among G , $\text{adist}(p', G)$ is less than or equal to $\text{maxdist}(e)^*m$. Moreover, $\text{maxdist}(e)^*m$ is less than or equal to $\text{adist}(q, G)$, so $\text{adist}(p', G) \leq \text{adist}(q, G)$. In other words, q is not the GNN of G . Therefore, we have Lemma 3 is correct. \square

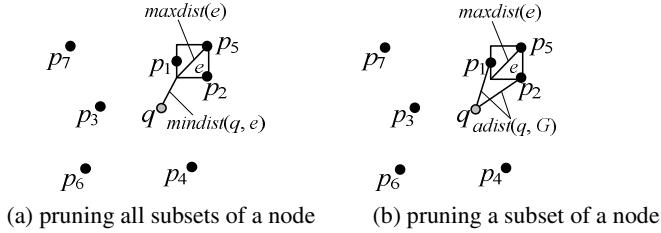


Fig. 2. Illustrate the pruning heuristics

Query Processing

In this subsection, we describe the query procedure of LRkGNN algorithm. LRkGNN first initialize some variables at line 1, and then inserts all entries of *root* node of R-tree into an auxiliary heap H (line 2). The *key* is the minimum distance between current entry e and q . LRkGNN utilizes Theorem 1 and Corollary 1 to stop the search (lines 5~7). The lazy technique reflects on lines 9~18. Line 12 progressively generates candidate subsets in ascending order of $\text{adist}(q, G)$. Line 13 discards candidate subsets by Lemma 2 and Lemma 3. Line 14 guarantees that LRkGNN only need to evaluate a few of subsets when progressively output subsets. LRkGNN processes current subset G , where the procedure *UpdateRlt* is used to update the query results and current best aggregate distance *best_kdist* (Lines 15~17). The intermediate entry is expanded by lines 20~21. Line 22 employs Lemma 2 and Lemma 3 to mark the *false alarms*. LRkGNN makes use of lines 23~29 to expand the subsets which contains the current entry e using a similar procedure of lines 15~17.

Algorithm 1. LRkGNN(root, P, q, m, k)

Input: The R-tree index over P , a query object q , two user specified parameters m and k
Output: the combinations which have q as their GNN

/* G_{rlt} : the set of query results; G_{rfn} : the set of refined subsets; H : auxiliary min-heap; H_a : min-heaps, sorted in ascending order of their distance from q . */

1. $G_{rlt} = \emptyset$, $G_{rfn} = \emptyset$, $H_t = \emptyset$, $H_a = \emptyset$, $\text{best_kdist} = +\infty$, initialize min-heap H accepting entries (e , key)
2. insert ($\text{root}(I)$, $\text{mindist}(e, q)$) into heap H
3. **while** (heap H is not empty) **do**
4. remove top entry e from H
5. **if** ($\text{best_hdist} \geq \text{best_kdist}$)
6. **if** (G_{rfn} is empty) **return** // Theorem 1
7. **if** (G_{rfn} isn't empty and $\text{best_rfndist} \geq \text{best_kdist}$) **return** // Corollary 1
8. **if** (e is a data object)
9. $H_t = H_t \cup e$
10. **if** ($C_{|H_t|^m} \leq k \leq C_{|H_t|+1}^m$) $H_a = H_t \cup H$ //applying (1)
11. **else continue**
12. **for** $\forall G$ formed by the entries in H_a **do** //in ascending order of $\text{adist}(q, G)$
13. **if** (G is marked as *false alarm*) **continue** //Lemma 2 and Lemma 3
14. **if** ($\text{adist}(q, G) \geq \text{best_kdist}$) **break**; //Lemma 1
15. **if** (G consists of data objects)
16. **if** (q is the GNN of G) $\text{UpdateRlt}(G, G_{rlt}, k, \text{best_kdist})$
17. **else** insert G into G_{rfn}
18. $H_t = \emptyset$, $H_a = \emptyset$
19. **else** //leaf node or intermediate node
20. **for** each data object or entry $e_i \in e$ **do**
21. insert e_i of e into heap H
22. mark \forall combination G of e pruned by Lemma 2~3 as *false alarm*
23. **for** \forall combination G of G_{rfn} including e **do**
24. remove G from G_{rfn}
25. **for** each updated combination G' of G **do** //replace e of G using e_i
26. **if** ($\text{adist}(q, G') \geq \text{best_kdist}$) **continue**
27. **if** (G' consists of data objects)
28. **if** (q is the GNN of G') $\text{UpdateRlt}(G', G_{rlt}, k, \text{best_kdist})$
29. **else** insert G' into G_{rfn}

3.2 Spatial Pruning LRkGNN Algorithm

Since LRkGNN need to invoke the MBM algorithm for each candidate subset, we present, in this subsection, an enhanced algorithm, called *Spatial pruning Reverse top- k Group Nearest Neighbor algorithm* (SLRkGNN), which takes advantage of *spatial pruning* method to identify or discard current candidate subset and does not has to invoke MBM algorithm to compute GNN.

In the sequel, we illustrate the intuition of our spatial pruning method by Figure 3. Specifically, let q be a query object and $G = \{p_1, p_2, \dots, p_m\}$ be a candidate subset which contains in the dataset P . In particular, we denotes the *pruning region* (circle) of object p_i , centered at point p_i and with radius $\text{dist}(q, p_i)$, as $PR(q, p_i)$. For any combination G , its pruning region is defined as the union of pruning region of all objects in G . Moreover, we define the intersection of pruning region for G as $\cap PR(q, G) = PR(q, p_1) \cap PR(q, p_2) \dots \cap PR(q, p_m)$.

By analyzing the cases in Fig. 3, we immediately obtain the following Lemma 4.

Lemma 4. (*RGNN Spatial Pruning*) Given a candidate subset G and a query object q . If there exists at least an object $p' \in P \setminus G$ which is contained in $\cap PR(q, G)$, G is not the result of RGNN query of q ; if there does not exist any objects contained in $\cup PR(q, G)$, G is the result of RGNN query of q ; otherwise, there exists at least an object contained in the region $R = \cup PR(q, G) \setminus (\cap PR(q, G))$, RGNN query only needs to search the objects among R to judge whether or not G has q as its GNN.

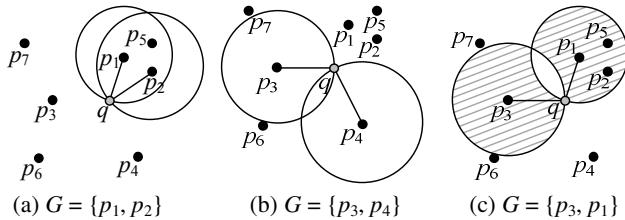


Fig. 3. Illustration of spatial pruning method in RkGNN query ($m=2$)

The importance of Lemma 4 is to enable RkGNN algorithm to shrink the search range when it cannot quickly judge whether current candidate subset G has q as its GNN. In practice, Lemma 4 saves much cost in distance computation and reduces many I/O accesses.

4 Experimental Evaluation

In this section, we experimentally evaluate the effectiveness of our proposed pruning heuristics and the performance of our developed algorithms for RkGNN query.

We use two real datasets, namely, *PP* and *NE*, from www.rtreeportal.org. We also create *Independent (IN)* and *Correlated (CO)* datasets with dimensionality $\text{dim} = 2$ and cardinality N in the range [20K, 100K] (*PP* is in the range [3K, 15K]). All datasets is normalized to range [0, 1]. Each dataset is indexed by an R-tree, with a page size of 4096 bytes.

We evaluate several factors, involving parameters k and m , and cardinality N . In each experiment, only one factor varies, whereas the others are fixed to their default values ($k=15$, $m=3$, $N=60K$). The *wall clock time* (i.e., the sum of I/O cost and CPU time, where the I/O cost is computed by charging 10ms for each page access, as with [1]), *the number of enumerating candidate subsets (NES)* which reflects the

performance of early stopping RkGNN computation, *the number of valid candidate subsets for GNN query (NVS)* which reports the capability of pruning heuristics (i.e., Lemma 1, Lemma 2, and Lemma 3, etc.), are used as the major performance metrics. Each reported value in the following diagrams in the average of 50 queries.

(1) The effectiveness of pruning techniques

This set of experiments aims at verifying the effectiveness of our proposed pruning heuristics. We vary k from 5 to 25, with m fixed at 3 in Figure 4, using four datasets, *PP*, *NE*, *CO* and *IN*. Evidently, the pruning techniques prunes a large number of candidate subsets (or say combinations), which validates their usefulness because *NES* and *NVS* both have a small value. In fact, a small value of *NES* verifies the efficiency of Theorem 1 and Corollary 1; meanwhile a small value of *NVS* confirms the efficiency of our proposed Lemma 1~Lemma 3. Although computing RkGNN is time consuming, our algorithm still obtains a higher performance owing to the adoption of early stopping technology which dramatically shrinks the search space. Similarly, Figure 5 and Figure 6 illustrate the prune efficiency of pruning techniques w.r.t. m and N , respectively, using real datasets and synthetic datasets, respectively. The diagrams confirm the observations and corresponding explanations of Figure 4.

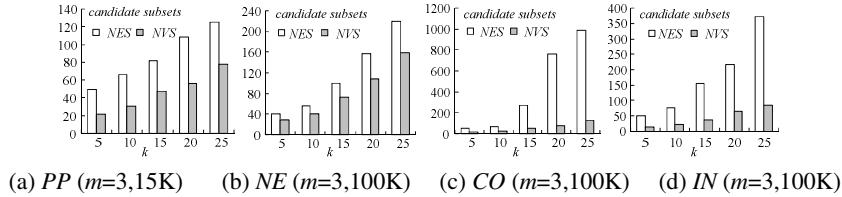


Fig. 4. Heuristic efficiency vs. k

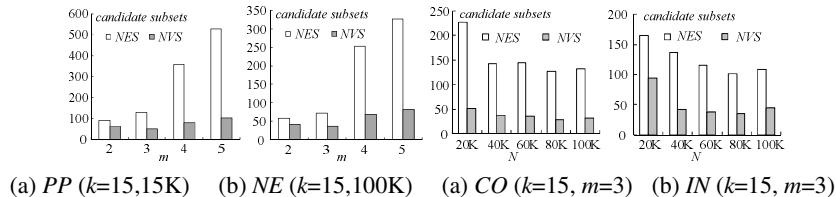
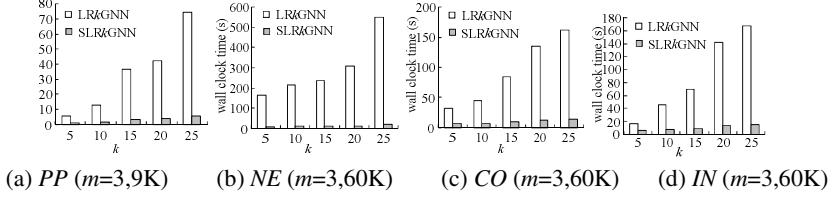


Fig. 5. Heuristic efficiency vs. m

Fig. 6. Heuristic efficiency vs. N

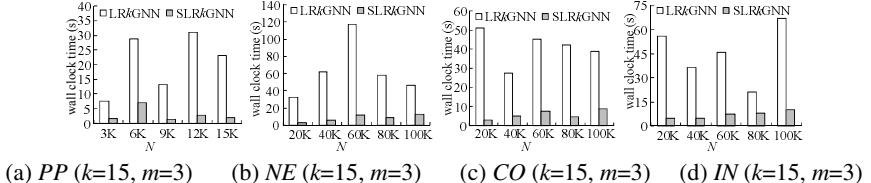
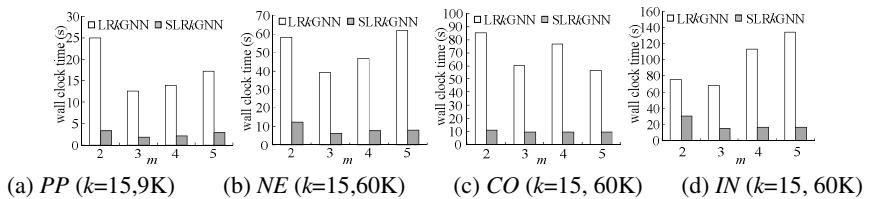
(2) The results on RkGNN queries

The second set of experiments studies the performance of our proposed algorithms in answering RkGNN queries. First, we explore the impact of k on algorithms, and the results are shown in Figure 7. Clearly, SLRkGNN outperforms LRkGNN in all cases because it integrates spatial pruning heuristics to prune unqualified candidate points. Furthermore, as k grows, the cost of LRkGNN and SLRkGNN slightly increase.

**Fig. 7.** RkGNN cost vs. k

Then, we evaluate the effect of N on the algorithms. Figure 8 depicts the cost as a function of N . SLR ^{k} GNN clearly gains a better performance than LR ^{k} GNN. When N increases, the cost of the algorithms does not always increase. This is because, on the one hand, the query objects is randomly choose on each test, and on the other hand, there is a big chance of finding some subsets which have q as their GNN in a big dataset than in a small dataset.

Finally, we inspect the effect of m on the algorithms, by fixing $k = 15$, $N=60K$ (except for $N=9K$ on dataset PP). Figure 9 shows the performance of algorithms as a function of m . Again, SLR ^{k} GNN outperforms LR ^{k} GNN in all cases. Moreover, although the number of candidate subsets dramatically increase as m increases, the cost of algorithms does not always grow. This is because on the one hand, Theorem 1, Corollary 1 and Lemma 1 prune most of the candidate subsets, and on the other hand, a big m brings more candidate subsets which have q as their GNN object.

**Fig. 8.** RkGNN cost vs. N **Fig. 9.** RkGNN cost vs. m

5 Conclusions

This paper firstly introduces and solves a new form of nearest neighbor queries, namely, *reverse top- k group nearest neighbor* (RkGNN) retrieval. We develop two

efficient algorithms, LR k GNN and S LR k GNN, using sorting and threshold technique, and some smart pruning heuristics to prune away most of unqualified candidate subsets. In the future, we intend to devise more efficient algorithm(s) for answering R k GNN queries by using the reuse technique. Another interesting direction for future work is to extend our approaches to tackle other variants of R k GNN queries, i.e., constrained R k GNN, the R k GNN in metric spaces, etc.

Acknowledgements. This work was supported in part by NSFC 61003049, ZJNSF LY12F02047 and LY12F02019, the Science and Technology Plan Fund of Jiaxing Municipal Bureau under Grant 2011AY1005, the Fundamental Research Funds for the Central Universities under Grant 2012QNA5018, the Key Project of Zhejiang University Excellent Young Teacher Fund (Zijin Plan).

References

1. Papadias, D., Tao, Y., Mouratidis, K., Hui, C.K.: Aggregate Nearest Neighbor Queries in Spatial Databases. *ACM Trans. Database Syst.* 30(2), 529–576 (2005)
2. Li, F., Yao, B., Kumar, P.: Group enclosing queries. *IEEE Trans. Knowl. Data Eng.* 23(10), 1526–1540 (2010)
3. Deng, K., Sadiq, S., Zhou, X., Xu, H., et al.: On Group Nearest Group Query Processing. *IEEE Trans. on Knowl. and Data Engineering* 24(2), 295–308 (2012)
4. Lian, X., Chen, L.: Probabilistic Group Nearest Neighbor Queries in Uncertain Databases. *IEEE Trans. on Knowl. and Data Engineering* 20(6), 809–824 (2008)
5. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate Nearest Neighbor Queries in Road Networks. *IEEE Trans. Knowl. Data Eng.* 17(6), 820–833 (2005)
6. Zhu, L., Jing, Y., Sun, W., Mao, D., Liu, P.: Voronoi-based Aggregate Nearest Neighbor Query Processing in Road Networks. In: *GIS*, pp. 518–521 (2010)
7. Xiao, X., Yao, B., Li, F.: Optimal Location Queries in Road Network Databases. In: *IEEE ICDE*, pp. 804–815 (2011)
8. Li, Y., Li, F., Yi, K., Yao, B., Wang, M.: Flexible Aggregate Similarity Search. In: *SIGMOD*, pp. 1009–1020 (2011)

Efficient Subsequence Search in Databases

Rohit Jain¹, Mukesh K. Mohania², and Sunil Prabhakar¹

¹ Department of Computer Sciences, Purdue University
305 N. University Street, West Lafayette, IN 47907

jain29,sunil@cs.purdue.edu

² IBM India Research Lab
New Delhi, India
mkmukesh@in.ibm.com

Abstract. Finding tuples in a database that match a particular subsequence (with gaps) is an important problem for a range of applications. Subsequence search is equivalent to searching for regular expressions of the type $. * q_1 . * q_2 . * \dots . * q_l . *$, where the subsequence is $q_1 q_2 \dots q_l$. For efficient execution of these queries, there is a need for appropriate index structures that are both efficient and can scale to large problem sizes. This paper presents two index structures for such queries based on *trie* and *bitmap*. These indices are disk-resident, hence can be easily used by large databases with limited memory availability. Our indices are applicable to dynamic databases, where tuples can be added or deleted. Both indices are implemented and validated against a naive approach. The results show that the proposed indices are efficient, having low I/O and time overhead.

1 Introduction

A subsequence is a sequence of symbols that is derived by removing some symbols from another sequence. A subsequence maintains the order among the remaining symbols. These symbols belong to a collection called *alphabet*. Subsequence search is equivalent to searching for regular expressions of the type $. * q_1 . * q_2 . * \dots . * q_l . *$, where the subsequence is $q_1 q_2 \dots q_l$. For example, “abc” is a subsequence of string “acbc”. Finding subsequences in a database is an important problem for many applications, *e.g.*, Automatic Speech Recognition (ASR) system or Optical Character Recognition (OCR) system. These systems can recognize a subsequence of the actual data. Hence, a subsequence search on the database of possible values can be useful.

As an example application, we consider two scenarios related to ASR. A voice call can be recorded and stored as an audio file (*e.g.*, a .wav file). When this file is transcribed into a text file, some noise is added during the transcription because many words are not recognized by the ASR system [1]. Similarly, when numbers are transcribed, many digits are missed during the transcription, but the ASR preserves the sequence of these numbers, *e.g.*, a phone number 9878281823 may be transcribed as 9878213.

Scenario 1: When a call (fixed line/mobile) is made, a CDR (Call Data Record) is generated by the switch that contains the information about “Who is calling whom”, time, duration of the call, etc. To track the activities of suspected individuals, the Police of ABC city has decided to record the calls made/received by these individuals. These

recordings may have important information, such as, the phone numbers of other potential associates, in addition to the person to whom the call was made. Based on these numbers, the police would like to trace all the calls made/received from these numbers. This can help the Police identify the social circle of these suspected individuals.

Scenario 2: A call is made to a call center. The customer gives her name and Social Security Number (SSN). Due to noise the call receiver does not understand some digits in the number. Even though a few digits are missing from the SSN, the receiver wants to pull the record from the database with the use of customer name and partial SSN.

In both the scenarios, it is desired that we get the tuples that have the transcribed text as a subsequence. The database of strings (phone numbers in the first case, and SSNs in the second case) is dynamic, *i.e.*, tuples can be inserted or deleted from the database. Since the database can be large, it is important that the index is disk resident and does not require the whole index to be stored in main memory.

In this paper we address the problem of efficiently finding subsequences in databases. We develop two index structures based on *trie* and *bitmap* to solve this problem. Our index based on *trie* uses an augmented suffix tree to encode strings in the database. Both indices are designed for large databases and are thus disk-resident and not limited by memory availability. The second index consists of a set of bitmaps over the data. Each search accesses only a subset of the bitmaps depending upon the query sub-sequence. To demonstrate the merits and demerits of these index structures we have implemented both index structures and analyzed them empirically.

The rest of the paper is organized as follows. In Section 2 we formally define the problem of subsequence search in databases. We explain our index structures in section 3 and 4. Experimental results are presented in Section 5. We summarize some related work in Section 6, and finally section 7 concludes the paper.

2 Problem Statement

We now formally define the problem of subsequence search. Consider a table T with n tuples and an attribute A of type string, *i.e.*, a sequence of characters. The characters that make up these strings are taken from a finite alphabet, Σ , of cardinality k . Furthermore, let m be the maximum length of the attribute for any given tuple. We are interested in the following regular expression match query. Given a sequence of l characters (from Σ), where $l \leq m$, find all tuples in T where this sequence occurs in the attribute A , possibly with intervening characters. Thus, given a query sequence: q_1, q_2, \dots, q_l , the goal is to identify all tuples that match the regular expression: $\{.*\} q_1 \{.*\} q_2 \{.*\} \dots \{.*\} q_l \{.*\}$. For example, given a subsequence “abc”, strings like “abcd”, “adbc”, “adbec” satisfy the query and should be returned. Table 1 summarizes different notations used throughout this paper.

One approach could be check each tuple individually, *e.g.*, using KMP [2]. However, such approaches will take at least order $O(n)$ time as each tuple has to be processed separately. To handle large databases, the index needs to be disk-resident. The index should also be able to handle dynamic databases where new tuples can be inserted and existing tuples can be deleted. In the further sections, we will describe our proposed index structures which solve this problem while being disk-resident. Our solutions can also handle dynamic databases.

Table 1. Symbols used in the paper

Symbol	Meaning
\sum_k	Alphabet from which characters are taken
n	Cardinality of alphabet $\sum (\mid \sum \mid)$
m	Number of tuples in table
l	Maximum length of attribute
$node(x, y)$	Number of symbols in query
	Node at level x and position y

3 Trie-Based Solution

Our first solution is based upon the following observation. Suppose the input subsequence consists of a, b, c (in that order). An intuitive solution for finding matching tuples is to process each tuple at a time. For a given tuple, we begin by finding the first occurrence of a . If none is found, we reject the tuple. If we find a then we search for the first occurrence of b in the remaining substring, and so on. This simple approach is guaranteed to be correct, however searching each tuple individually is very inefficient. In order to speed up the search, we exploit a *trie* index over the table. Trie structures have been popular for string searching. We modify this structure to make it suitable for our problem. In the following subsections, we will describe the structure of our trie-based index and how search and updates are done.

3.1 Structure

Each node in the index tree represents a symbol from the alphabet. The edges in the tree represent the sequence in which two symbols occur in the database. Consider a sample table with two attributes: a tuple id and a string attribute with characters from the set $\{a, b, c\}$, shown in Table 2. Given this data, we construct our index tree as shown in Figure 1. Each path from the root to a leaf represents a tuple in the database. Each node is labeled with a symbol from the tuple.

Level of a node is defined by its depth in the tree. The root has level 0. Each node also has a *position* assigned to it which represents its position in that level, *i.e.*, its position in the insert sequence for that level. E.g., in Figure 1, at level 2, nodes are inserted in this order: ‘b’ of tuple 1, ‘a’ of tuple 2, ‘c’ of tuple 3, ‘b’ of tuple 4, ‘c’ of tuple 5, and then ‘c’ of tuple 6. Positions are assigned accordingly. We represent a node as $node(x, y)$, where x represents the level and y represents its position. Node positions are also used when storing data on the disk.

Definition 1. *Fresh-occurrence* of a symbol in a subtree is defined as occurrence of a symbol in the subtree when none of its ancestors in the subtree has the same symbol except for the subtree root.

In figure 1, fresh occurrences of ‘c’ at the root are $node(2, 3)$, $node(2, 6)$, and $node(1, 3)$. Note that $node(2, 5)$ is not included since $node(1, 3)$ is its ancestor and has the same symbol. Similarly, fresh-occurrence of ‘c’ at $node(1, 3)$ is $node(2, 5)$.

Table 2. Example Table

TupleID	Attribute
1	ab
2	ba
3	bc
4	cb
5	cc
6	acb

Table 3. Example Table 2

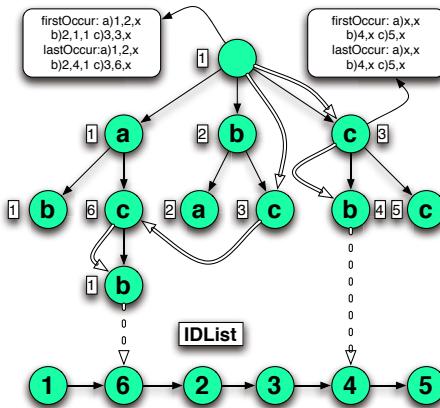
TupleID	Attribute
1	23332333
2	25382232
5	33232323
3	58382533
4	73358231

Each node is augmented with two 2-dimensional arrays, namely *firstOccur* and *lastOccur*, which stores the first and last fresh-occurrence of each symbol at each level in its subtree, respectively. Notice that there can be many fresh-occurrences of a symbol at the same level. However, *firstOccur* and *lastOccur* only track the first and last fresh-occurrences of them. In Figure 1, at root, *firstOccur* of ‘c’ would be *node(1, 3)* at level 1 and *node(2, 3)* at level 2, and, *lastOccur* will be *node(1, 3)* and *node(2, 6)* at level 1 and 2 respectively. Each node also maintains *next*, a pointer to the next fresh-occurrence of the same symbol in the same level. This is useful to traverse all the nodes with the same symbol from *firstOccur* to *lastOccur*. Hence, *firstOccur*, *lastOccur* and *next* are sufficient to find all the fresh-occurrences of a symbol at a given level. At root, if we want to traverse all the fresh-occurrences of ‘c’ at level 2, we do it like this: *firstOccur* of root tells ‘c’ occurs at *node(2, 3)*, *lastOccur* tells that ‘c’ occurs at *node(2, 6)*. So, we go to *node(2, 3)*, then we go to *node(2, 3).next* which is *node(2, 6)*. Since, this is the last fresh-occurrence we stop here. Table 4 summarizes the fields maintained by a node. From figure 1, we see that *firstOccur* and *lastOccur* at the root of the tree stores the first and the last occurrences of each symbol at all three subsequent levels. Similarly, *node(1, 3)* stores data for two levels. By choosing such data structure, we ensure that at each level, the size of each node is constant. This helps in the storage and retrieval of node data on disk.

At each node, we also store a list of tuple ids that fall in the subtree rooted at that node. For this, we maintain a link-list of IDs. Nodes store *startTupleID* and *endTupleID* pointers to this list. For example, at *node(1, 2)*, *startTupleID* and *endTupleID* will be 2 and 3 respectively. Now if the regular expression is satisfied at this node, we simply return all the nodes from 2 to 3 in the IDList. This means that all tuples ids from 2 to 3 in the list have common prefix, which is ‘b’.

3.2 Search

The search for tuples matching the query sequence “acb” begins at the root. At the root, we find that ‘a’ is present at *node(1, 1)*, and *node(2, 2)*. Thus, all the tuples in these subtrees will satisfy $\{.\}^*a\{.\}^*$. Now, we search $\{.\}^*c\{.\}^*b\{.\}^*$ in the subtrees rooted at these nodes. We find that ‘c’ is present at *node(2, 6)* only. Consequently, we search for ‘b’ in this subtree. We find that ‘b’ is present only at *node(3, 1)*. All tuple ids that are stored in the subtree rooted at this node constitute the answer to the query. In this case, it will be the tuple with id 6. Algorithm 1 explains the search process formally.

**Fig. 1.** Example tree based on table 2**Table 4.** Data stored at each node

Data Item	Description
ID startTupleId	Tuple id of first tuple under the node.
ID endTupleId	Tuple id of last tuple under the node.
Var Label	symbol stored at the node, Null for the root
Node[][] firstOccur	Pointers for first fresh-occurrence of each symbol in further levels
Node[][] lastOccur	Pointers for last fresh-occurrence of each symbol in further levels
firstOccur[x][y]	First fresh-occurrence of x at level y
lastOccur[x][y]	Last fresh-occurrence of x at level y
Int level	Node's level
Node next	next node in the level list
Int position	node's position
Node parent	Parent position

To explain the use of *next*, we consider a search for subsequence “cb”. At root, we check *firstOccur*, and find that *c* is present at *node(1, 3)* and *node(2, 3)*. Figure 1 explains step by step how the search is conducted. First, we go to *node(1, 3)* and search for ‘b’. We find that ‘b’ is present at *node(2, 4)*, so we output ID 4. Then we search for ‘b’ at *node(2, 3)*. We find that there is no ‘b’ in its subtree. We check *lastOccur* of root and find that *node(2, 3)* was not the last fresh-occurrence of ‘c’ at that level, so we go to the *next* of *node(2, 3)*, i.e., *node(2, 6)*. We search for ‘b’ at this node and find a ‘b’ at *node(3, 1)*. So, we output ID 6. Since this is the last occurrence of ‘c’, we stop here. Notice that we never searched for ‘c’ at *node(2, 5)*. This is because, search at *node(1, 3)* was sufficient as that was first occurrence of ‘c’. As data size increases, such cases of pruning of nodes will increase, improving performance of this index.

Due to space limitations, we omit the algorithms for insert and delete of a tuple in the index.

Algorithm 1. SEARCH (Var[] *regex*, Node *root*)

```

1: List tuples;
2: if regex.length = 0 then
3:   return tuples_in_subtree(root)
4: end if
5: for i = 0 to m – root.level do
6:   if root.firstOccur[regex[0]][i] ≠ -1 then
7:     node ← root.firstOccur[regex[0]][i]
8:     repeat
9:       tuples.union(SEARCH(regex + 1, node))
10:      node ← node.next
11:    until node ≠ root.lastOccur[regex[0]][i]
12:   end if
13: end for
14: return tuples;

```

3.3 Analysis

Space: There are $m + 1$ levels in the tree. *firstOccur* and *lastOccur* of each node are 2 dimensional arrays with each row representing each symbols from the alphabet and each column representing different levels in the subtree. Thus, each node will have a maximum size of $O(mk)$. In the worst case, there can be $O(nm)$ nodes, where n is the number of tuples in the table. So, in the worst case, the space required would be $O(nm^2k)$. However, in practice, many tuples will share common prefixes. As tuples start sharing common prefixes, space requirement will decrease. When all possible symbol combinations are added to the database (*i.e.*, a full tree), there will be a total of $O(n)$ nodes in the tree. Hence, the space requirement will be $O(nmk)$.

Search Time: In worst case, the search time would be $O(nl)$. This is because at each level there can be at most $O(n)$ nodes, and we have to go through $O(l)$ levels to satisfy the subsequence.

4 Bitmap-Based Index

Tree based index structures have the property of representing data in a concise way which makes it fast to search and access the desired data. However, these indices can have complex ways to traverse and also to store data on disk. Bitmap indices enjoy a very simplistic structure which are fast and easy to implement. Even though bitmap indices are very simplistic, one should be careful in designing these indices correctly.

As a naive attempt at using bitmaps to solve this problem, consider a simple set of bitmaps $\{B_{c_1 c_2} | c_1 \in \sum, c_2 \in \sum\}$. In other words, we maintain k^2 bitmaps each of which corresponds to the occurrence of a pair of characters in order. The i^{th} bit in the bitmap B_{ab} is 1 if the i^{th} tuple contains ‘a’ followed by ‘b’, and otherwise it is 0. Given this set of bitmaps, we may try to find all tuples matching the query “abc” by taking the bit-wise AND of the three bitmaps: $B_{ab} \odot B_{bc} \odot B_{ac}$. However, this can lead to incorrect results as seen from the tuple: “bacb”. It is clear that this sequence satisfies the conditions for the three bitmaps, but it does not contain the query sequence.

In this section we propose an alternative solution to the subsequence search problem using bitmap indices. The next subsection describes our bitmap based index and its use to search for subsequences.

4.1 Proposed Solution

Since space is not a constraint, we propose to maintain a large number of bitmaps: $\{B_{c_1 c_2 i j} | c_1, c_2 \in \Sigma, 0 \leq i < j < m\}$. Each bitmap corresponds to the occurrence of a pair of characters at given positions, where c_1 and c_2 are characters from the alphabet Σ , and i and j are possible locations, where $0 \leq i < j < m$. If in a tuple, characters c_1 and c_2 appear at positions i and j respectively, then the bit corresponding to that tuple in index $B_{c_1 c_2 i j}$ is set to 1, otherwise it is 0. Thus, we have $k^2 m^2$ bitmaps, each of length n bits. Note that while this seems to be a large number of bitmaps, we expect that there will be many zeros in the bitmaps which could then be effectively compressed using an encoding scheme such as [3].

To see how these bitmaps can be used to find the answer to a query, consider the query sequence “abc”. It is clear that any sequence that contains this sequence must have bits 1 at least in the following two bitmaps: B_{abij}, B_{bcjk} where $0 \leq i < j < k < m$. Thus, to identify the set of tuples that satisfy the query we need to take a bitwise-AND of pairs of bitmap for a given value of i, j , and k and then take the bitwise-OR of each of these results. In other words, we compute the following bitmap:

$$\bigoplus_{0 \leq i < j < k < m} (B_{abij} \odot B_{bcjk})$$

All tuples that correspond to the bits set to 1 in this result are tuples that must contain the query sequence. For larger sequences, we apply the same technique, *i.e.*, in general to search for the query sequence q_1, q_2, \dots, q_l , we compute the following bitmap:

$$\bigoplus_{0 \leq i_1 < i_2 < \dots < i_{l-1} < i_l < m} (B_{q_1 q_2 i_1 i_2} \odot B_{q_2 q_3 i_2 i_3} \odot \dots \odot B_{q_{l-1} q_l i_{l-1} i_l})$$

The number of AND and OR operations is given by: ${}^m C_{l-1}$ OR operations and ${}^m C_{l-1}(l-1)$ AND operations.

For example, for Table 3, the bitmap indices will look something like this:

	1	2	5	3	4		1	2	5	3	4
<2, 3, 1, 2>	1	0	0	0	0		<3, 3, 1, 2>	0	0	0	1
<2, 3, 1, 3>	1	1	0	0	0		<3, 3, 2, 3>	1	0	0	0
<2, 3, 1, 4>	1	0	0	0	0		<3, 3, 3, 7>	1	1	1	0
<2, 3, 1, 6>	1	0	0	0	0		<3, 3, 4, 6>	1	0	0	1
<2, 3, 1, 7>	1	1	0	0	0		<3, 3, 5, 6>	0	0	0	0
<2, 3, 3, 4>	0	0	0	1	0		<3, 3, 5, 7>	0	0	0	0
.....										

Searching for “233” will first require to find all possible combinations for positions of “2”, “3” and “3” in the attribute. Among different combinations for positions of symbols, we have (1, 2, 3), (1, 3, 7), (3, 4, 6), and so on. Now, we first take ANDs:

(1, 2, 3) : <2, 3, 1, 2> AND <3, 3, 2, 3> : 10000
 (1, 3, 7) : <2, 3, 1, 3> AND <3, 3, 3, 7> : 11000
 (3, 4, 6) : <2, 3, 3, 4> AND <3, 3, 4, 6> : 00010

Next the OR of these bitmaps is computed yielding 11010. This means that the tuples with TupleID 1,2, and 4 satisfy the regular expression. Since we have not calculated all the combinations, we can not say anything about other tuples. Upon completing the calculation, we will find that the output bitmap will be 11110, hence tuple 5 does not satisfy the regular expression. Therefore, the output IDs will be 1, 2, 5, 3.

4.2 Analysis

Space: As explained before, there will be $O(m^2k^2)$ bitmaps, and since each bitmap stores 1 bit for each tuple, the size of each bitmap will be $O(n)$. Hence, the space required is: $O(nm^2k^2)$. To search for a subsequence of length l , we will need to use $(l - 1) \times (m - l + 1) \times (m - l + 2)/2$ bitmap indices.

Time: For one AND or OR operation between 2 bitmaps, we need $O(n)$ time. Now, for a particular combination of locations, we do $l - 1$ AND operations. And we take OR of each such combination of locations. Now there are mC_l such combinations. Therefore, time required is $O(n \times l \times {}^mC_l)$.

5 Experimental Evaluation

To establish the effectiveness of the proposed indices, we implement both indices and a naive approach. To evaluate the efficiency of the proposed indices, we run three sets of experiments. First set of experiments demonstrates the disk usage and index creation time of the indices. Secondly, we demonstrate the effect of data set size on the time and disk I/O required in subsequence search. And finally, we demonstrate the effect of query size (subsequence length, l) on search time and disk I/O.

In all experimental results, time is reported in milliseconds and disk usage is reported in megabytes. We implement the index structures in C. The experiments are run on an Intel Xeon, 3.0 GHz machine with 2 x 512MB (266MHz) RAM and PATA, 7200 RPM, 8.5ms average read time disks running Linux.

Naive Approach: In naive approach, we take each tuple one by one, and we parse it sequentially. For example, when searching for $\{.\}a\{.\}b\{.\}c\{.\}$, we take a tuple, and search for a in it. If we do not find a in the tuple, it is discarded else we search for b in the remainder substring, and so on.

5.1 Setup

Storage Organization: We now discuss how these indices are stored on disk. The node data for each level of the tree are stored in a separate file. To add a new node at a given level, it is appended to the corresponding level file. If the data of a node need to be modified, we modify the entry in place in the file. We can know the node's start

Table 5. Dataset for experiments

Dataset Size	T1 Size	T2 Size	T3 Size
1000	10	10	10
10000	10	10	100
100000	10	100	100
1000000	10	100	1000
10000000	100	100	1000
20000000	100	100	2000
30000000	100	100	3000
40000000	100	100	4000

position in the file simply by knowing i and level. Since each node, $node(x, y)$, can be identified by values of x and y , all the pointers are actually stored in the form of node position at the corresponding level, i.e., $firstOccur$, $lastOccur$, $next$ and $parent$ keep positions of nodes in it's level. as that is always known. IDList is stored in a separate file. Whenever a new ID needs to be added, it is appended to the end of the file. Again, location of ID in IDList file is used as pointer values. Using this approach, we do not need to keep the whole index structure in memory. We can read a node on a need to know basis. For the bitmap index, each bitmap is stored in a separate file. For every 8 new tuples, 1 byte is added to each of the bitmap files. For naive approach, we store data in text form. Each tuple is followed by new line.

Data Sets: We use a synthetic dataset for the experiments. To simulate US phone numbers (of the form xxx-xxx-xxxx), we generate 10 digit numbers randomly. We generate them in a set of 3, 3 and 4 digits. In other words, we generate 2 sets of 3 digit numbers, and 1 set of 4 digit numbers (namely T1, T2 and T3). A phone number is generated by picking one number from each set. All the phone numbers generated are unique. Please note that both indices will work even if there are duplicate entries. Table 5 shows size of T1, T2 and T3 for each dataset.

Buffer Manager: In order to simulate the effect of buffering, we use a buffer manager of size 1MB. For *trie*-based index, only buffer size of data is kept in main memory for each level. This means, on an average, memory used will be approximately $(m + 1)$ times the buffer size. For bitmap and naive approach, cache hits will not occur these approaches as data is read sequentially. Thus, we simply assign $(m+1)$ buffer size memory to them. Buffer Manager estimates the amount of I/O done to read index data from the disk to execute a subsequence search.

Queries: For each dataset, we pick a number, and search for its subsequences. The subsequence length varies between 6 and 10. There are 2 variables we change when running the queries, a) Dataset size, i.e., number of tuples b) Regular expression length.

5.2 Results

We now discuss the results of our experiments. As mentioned before, we have mainly three set of experiments to evaluate the fidelity of our proposed index structures. Now, we describe the setup of each set of the experiments and the results of the experiments.

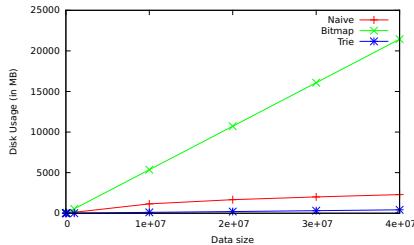


Fig. 2. Disk space required to store index

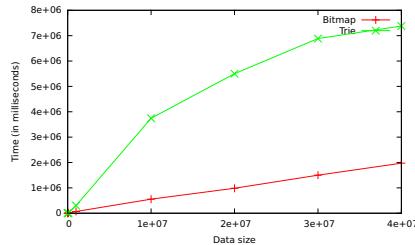


Fig. 3. Total time required to create index

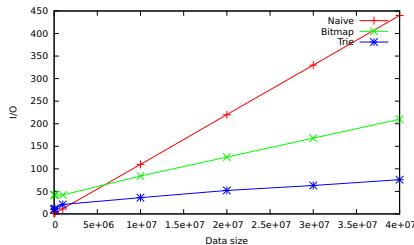


Fig. 4. I/O vs data size ($l = 8$)

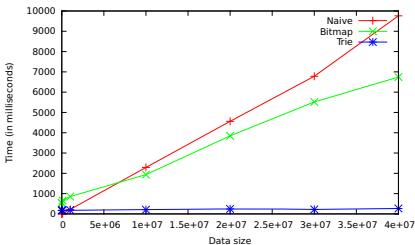
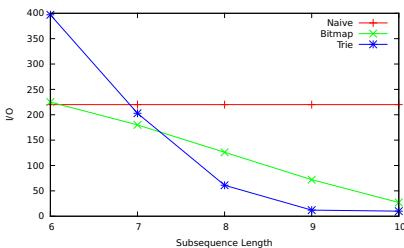
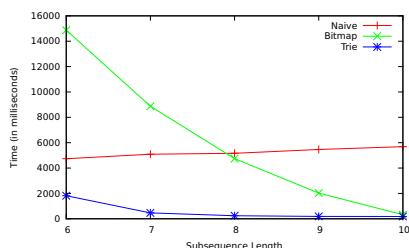


Fig. 5. Time vs data size ($l = 8$)

Index Size and Create Time: Figure 2 shows the effect of dataset size on the size of index structure. As the dataset size increases, the trie-based index becomes *saturated*, i.e., the index tree tends towards a full binary tree. As a result, number of new nodes required to insert a new tuple decreases as the dataset size increases. Thus, the size of the index tree does not increase rapidly. On the other hand, the bitmap index size increases linearly with the dataset size. This is because the size of each bitmap depends only on the number of tuples in the dataset. Figure 3 shows the effect of number of tuples on the total time to create the indices. Since there is no index structure for the naive approach, we do not show it in this figure. Even though the increase in the data size decreases the number of new nodes required per insert, existing nodes will still have to be updated to accommodate new nodes (for example, *firstOccur* and *lastOccur* of existing nodes will be modified). This results in higher creation time for the index tree. However as the dataset size increases, time required per insert decreases. The time required for creating the bitmap index is much less due to its simplistic structure.

Effect of Dataset Size on Search: In order to understand how our proposed index structures scale for large databases, we demonstrate the effect of dataset size on search time and disk I/O. We increase the dataset size to as large as 40 million so that we could evaluate the indices for large databases. For each dataset, we search for a subsequence of length 8. Figure 4 and 5 demonstrate the effect of the dataset size (number of tuples) on disk I/O and query time required for searching a subsequence, respectively. Bitmap indices increase linearly with the dataset size. Thus, disk I/O and search time both increase linearly. Naive approach shows similar trend. However, the *trie* based data structure does not exhibit a similar growth rate. The *trie* structure saturates as dataset

**Fig. 6.** I/O vs query length ($n = 20000000$)**Fig. 7.** Time vs query length ($n = 20000000$)

size increases. Thus, the number of extra nodes required to read does not increase as much, resulting in small disk IO and search time.

Effect of Query Size on Search: In many applications, accuracy of transcribed sequence can be different. Hence, its important to understand how these indices behave when subsequence length changes. Figure 6 and 7 show the effect of subsequence length on disk I/O and time, respectively. As discussed before, a query of length l requires $(l - 1) \times (m - l + 1) \times (m - l + 2)/2$ bitmaps. Thus disk I/O in bitmap index is maximum for regular expression length 5. After this overhead keeps decreasing. As expected, for the naive scheme, time and I/O are largely unaffected by the subsequence length. For trie-based index, as the subsequence length increases, the movement between different levels is constrained. For example, when searching for 12345, we look for “1” at levels from 1 to 5. On the other hand, if we search for 1234567, then “1” will be searched at levels 1, 2 and 3. Hence, the disk overhead is expected to decrease as subsequence length increases. These expectations are met in the experiment results.

From our experimental results, we see that both indices based on bitmap and trie are effective for large subsequence searches. The index based on trie is orders of magnitude effective even for short subsequences. Our indices scale for large databases while using acceptable amount of disk.

6 Related Work

There are two broad approaches for processing regular expression queries. One is to create a deterministic finite automaton and pass each tuple through it. This approach does not scale as each tuple has to be processed separately. Also the number of state in the DFA can be exponentially large. Second approach is to preprocess the data ([4–6]) and construct an index structure, which is used to process regular expression queries.

The *trie* datastructure[7] has been a popular index structure for string searching [4, 8, 9]. [4] showed how regular expressions can be searched using the Patricia Tree. The time complexities of their index are quite efficient. However, this index is suitable only if the entire trie fits into main memory, which may not be feasible for large databases. Another problem with this index is that it works only for a *static* databases. Our indices are both efficient and can handle *dynamic* databases as well.

[10] extends the pattern expressions to *parameterized pattern queries*. The paper introduces the concept of variables to express regular expressions, which makes it more

concise and allows additional constraints. [11] proposes an R*-tree based index for fast subsequence search in time-series databases. They use a different definition of subsequence matching. Given two strings, S and Q , the subsequence matching problem is that the distance between S and Q should be within a static value ϵ .

FREE [12] uses a multigram index for searching regular expressions. It selects frequent grams, and indexes them. This index is used to find tuples which may potentially satisfy the regular expression, and then those tuples are searched. However, For this technique to yield good results, the grams should be powerful, which may not be the case in many applications like ASR.

The *RE-Tree* [6] tries to address the opposite problem. Given a database of many REs and a string, the proposed solution finds all REs that satisfy the given string. RE-tree is a height balanced tree in which each node stores an automaton (number of states in these automata are bounded).

7 Conclusion

In this paper, we considered the problem of searching subsequences in databases. We showed that the problem is important for many applications. We presented two disk-based index structures for this problem which scale to large databases. These indices not only work for static databases, they work for dynamic databases as well which allows inserts and deletes of tuples. We evaluated the efficiency of these indices by implementing them and comparing them on various aspects. We showed that our index structures are efficient and scale to large databases. We also showed that our indices can be efficiently stored on disk.

Acknowledgements. The work in this paper is supported by National Science Foundation grants IIS-1017990 and IIS-09168724.

References

1. Subramaniam, L.V., Faruque, T.A., Iqbal, S., Godbole, S., Mohania, M.K.: Business intelligence from voice of customer. In: International Conference on Data Engineering (2009)
2. Knuth, D.E., Morris, J., Pratt, V.R.: Fast pattern matching in strings. SIAM Journal on Computing 6, 323–350 (1977)
3. Antoshenkov, G.: Byte-aligned bitmap compression. In: Conference on Data Compression (1995)
4. Baeza-Yates, R.A., Gonnet, G.H.: Fast text searching for regular expressions or automaton searching on tries. J. ACM 43(6), 915–936 (1996)
5. Manber, U., Baeza-Yates, R.: An algorithm for string matching with a sequence of don't cares. Information Processing Letters 37(3), 133–136 (1991)
6. Yong Chan, C., Garofalakis, M., Rastogi, R.: Re-tree: An efficient index structure for regular expressions. The Very Large Databases Journal 12(2), 102–119 (2003)
7. de la Briandais, R.: File searching using variable length keys. In: AFIPS Western JCC, San Francisco, Calif., pp. 295–298 (1959)
8. Ukkonen, E.: On-line construction of suffix trees. Algorithmica 14(3), 249–260 (1995)

9. McCreight, E.M.: A space-economical suffix tree construction algorithm. *J. ACM* 23(2), 262–272 (1976)
10. du Mouza, C., Rigaux, P., Scholl, M.: Parameterized pattern queries. *Data Knowl. Eng.* 63(2), 433–456 (2007)
11. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. *SIGMOD Rec.* 23, 419–429 (1994)
12. Cho, J., Rajagopalan, S.: A fast regular expression indexing engine. In: International Conference on Data Engineering, pp. 419–430 (2002)

A Novel Data Broadcast Strategy for Traffic Information Query in the VANETs^{*}

Xinjing Wang¹, Longjiang Guo^{1,2,✉✉}, Jinbao Li^{1,2}, Meirui Ren^{1,2}

¹ School of Computer Science and Technology, Heilongjiang University, China

² Key Laboratory of Database and Parallel Computing, Heilongjiang, China

longjiangguo@gmail.com

Abstract. This paper proposes a data broadcast strategy for traffic information query. Traffic information query is an important application in VANETs. Real-time traffic information query makes users to select path in a short time. By this data broadcast strategy, traffic information query can be quick and accurate after data is broadcast and forwarded. According to the restricted condition in the strategy, the vehicles broadcast their packets, and update, combine-forward or drop the data after receiving them in VANETs. In this way, more vehicles can receive data from source region. This algorithm can reduce the redundant data and control the communication congestion effectively. This paper adopts NS2 and SUMO as simulation platform. Simulation results show that, data collision rate is no more than 30%, and information area coverage rate reaches more than 80% with the strategy proposed in this paper after setting proper parameters.

1 Introduction

The communication framework built by Vehicular ad hoc Networks(VANETs) and wireless communicating technology is one of the important foundation of the Intelligent Transportation System(ITS). The wireless communicating technology of VANETs can implement the applications in transportation field such as traffic accident prediction, driving assistance, traffic information query, besides can provide some services such as data download, vehicular entertainment, and

* This work is supported by Program for New Century Excellent Talents in University under grant No.NCET-11-0955, Programs Foundation of Heilongjiang Educational Committee for New Century Excellent Talents in University under grant No.1252-NCET-011, Program for Group of Science and Technology Innovation of Heilongjiang Educational Committee under grant No.2013TD012, the Science and Technology Research of Heilongjiang Educational Committee under grant No.12511395, the Science and Technology Innovation Research Project of Harbin for Young Scholar under grant No.2008RFQXG107, 2009RFQX080 and No.2011RFXXG014, the National Natural Science Foundation of China under grant No.61070193, 60803015, Heilongjiang Province Founds for Distinguished Young Scientists under Grant No.JC201104, Heilongjiang Province Science and Technique Foundation under Grant No.GC09A109.

✉ Corresponding author.

data query combined with WiFi. It can be foreseen that VANETs not only will provide broader space for the traffic intelligent and humanized improvement, but also will be convenient for people travel.

The real-time traffic information can help improve the traveling efficiency and reduce the vehicle congestion, thereby driving distance and time as well as energy consumption. The purpose of the proposed strategy for traffic information query is to locally containing more useful data before query. To achieve the purpose, we propose a broadcast algorithm combining with probability and random delay, and put forward the definition of influence scope to reduce the data redundancy again. In addition, the proposed data integration idea combines the similar traffic information to a new data, in order that data are more valuable, and more conducive for future inquiries. Verified by simulation, compared with common broadcast algorithm Epidemic, the proposed strategy improves remarkably in data collision rate, update degree, and information area coverage rate.

The rest of this paper is organized as follows. In the next section, we review the related works. We discuss vehicle broadcast and forward algorithm particularly in Section 3. Analytical performance study and simulation results are presented in Sections 4. Finally, we conclude this paper in Section 5.

2 Related Work

Data management in VANETs is a popular research issue at present, and so far there are a lot of research achievements in many aspects about this issue. [1] studies the data naming issues during the broadcast, and [2] focuses on the issue of data signatures. [3] analyzes data throughput and delivery delay in VANETs in detail. In this paper we study another data management issue, which is data broadcasting in VANETs. Besides the broadcast data is special, which is the urban traffic information data.

The wireless mobile network broadcast technology has matured, and nowadays broadcast algorithms on wireless mobile networks are also endless. The broadcast using Random Way Point model is relatively simple. The general study is theoretically analysis on its performance[4]. While heuristic routing protocol WHIRL[5] and the routing algorithm on large scale LAMMAR[6] use RGPM as their mobile model. At present most mobile models used for broadcast algorithms are these both, while VANETs belongs to opportunity network, unlike other network nodes which keep delivering data outward, the transmission process of it's nodes is store-carry-forward, which also leads to the algorithms in other networks are not suitable to be used for VANETs directly. In this paper, we study the broadcast algorithm supporting traffic information query in VANETs, and the mobile model used for it is urban traffic scenario, so we can't use the existing broadcast algorithms in other mobile model. We need to design an algorithm adjust to VANETs characteristics.

So far most researches about VANETs are routing algorithm, and the majority of which assume the senders have got the knowledge of the location of the destinations. About the broadcasting issues, some focus on the security issues when the

data broadcasting such as[7], some like[8] analyze data broadcast protocol on MAC layer ensuring data security in detail. There are relatively small work on pure data broadcasting currently, since for the ordinary data, the efficiency of Epidemic[9] has been sufficient to meet. Data whereabouts in the broadcast algorithm aren't clear destinations, but the data is territorial. Its goal is collect traffic information as much as possible for later data query, so the key point of this paper is how to collect more data with control network congestion effectively as premise.

3 Vehicles Broadcasting/Forwarding Algorithm

3.1 Issue Description

The background of the proposed strategy is urban traffic, and the mobile scene is vehicles moving in the urban streets. To make the issue simplify, denote A as the sum of areas of all the roads. There are N vehicles in the region, the average length of each street being L , the average of the streets being D , the vehicular communication range being R , and the vehicular average density per street being ρ . Since querying the real-time traffic information need to know the traffic information of roads in a short time, we need set a time period T . The discussed traffic information is divided into two kinds, unobstructed and jam.

To get the data quickly in the urban traffic mobile model, One of the methods is broadcasting data periodically before the inquiry, in order to find the desired data locally, reduce query delay, and improve query efficiency. So at data broadcast stage, every vehicle has to try to collect useful data and drop the redundant data. After a vehicle produces a data packet containing self-information, it will broadcast its packet; when a vehicle receives a packet, it will judge whether the packet is value to store and forward. If each of the vehicles broadcast and forward packets unlimited, it will lead to many problems such as channel competition, network congestion and overissue data.

Next we will introduce the solutions of the problem.

3.2 Related Data Structure and Operation

Data Structure of Packets. This paper focuses on the traffic information query, the data structure of the data packet stored in Vehicle i is represented by P_i , which includes center coordinates of influence scope covered by the packet (x, y) , coordinates of the vehicle (x_0, y_0) , transmit time of the packet t , velocity of the vehicle v , the direct vector (a, b) and affecting distance d . The definition of influence scope and affecting distance are introduced in next section. In the following content, denote $P_i.*$ as the variable in the packet P_i , for example, the center coordinate influence scope covered by P_i is $(P_i.x, P_i.y)$. Wherein the two coordinates are geographical absolute coordinates, which can be obtained by the GPS positioning. Both of the coordinates' initial value are coordinates of the vehicle when a data packet is generated. Once the packet is forwarded, the center coordinates in the packet will change to the center coordinates of the influence scope presented by

the packet, but the coordinates of the vehicle will change with the movement of the vehicle. The transmission time is the moment that the vehicle sends its a packet. Each vehicle has a storage area called the local memory, in which memory structure is a data table, and each of the record in which is a packet.

Related Definition

Definition 1. Influence Scope, Affecting Distance. Suppose vehicle i is center, build a rectangle whose length is $2d$ along \vec{v}_i direction and width is D along \vec{v}_i vertical direction. We call the rectangle as the influence scope for i , and d as affecting distance of i .

For example, Fig.1 shows the influence scope of Vehicle a . According to the Fig.1, Vehicle b and Vehicle e are both in the influence scope of Vehicle a . The packets have to contain the vehicle's coordinates and center coordinates represented by the packet, and the influence scope of the packet is a rectangle, the center of which is the center coordinates of the packet.

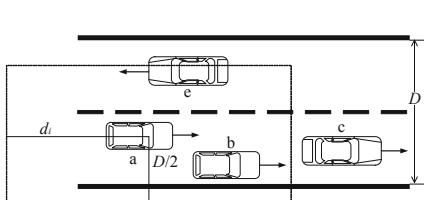


Fig. 1. Straight Road

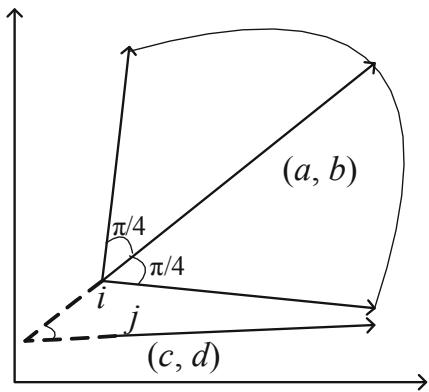


Fig. 2. Traveling Direction

Definition 2. Traveling Direction. Vehicle i produces the direction vector (a, b) on its direction lane in the time of Δt , according to which as a benchmark turning $\frac{\pi}{4}$ respectively in the clockwise and counterclockwise, and the resulting angle range is the traveling direction of the Vehicle i . Suppose Denote that Vehicle j produces the direction vector (c, d) on its direction lane in the time of Δt . If ϕ_{ij} that the angle between (a, b) and (c, d) meets that $-\frac{\pi}{4} < \phi_{ij} < \frac{\pi}{4}$, then we call that the traveling direction of Vehicle i and Vehicle j are the same.

Wherein the angle of the direction vectors of the two vehicles is

$$\phi_{ij} = \arccos \frac{(P_i.a, P_i.b) \cdot (P_j.a, P_j.b)}{|(P_i.a, P_i.b)| |(P_j.a, P_j.b)|} \quad (1)$$

Definition 3. Information Area. The coverage area union set of influence scope in the packet stored locally.

Judgment Conditions of Forwarding Packets. When a vehicle receives a packet, it will determine to drop or store and forward the packet according to the following conditions.

Proposition 1. *The necessary and sufficient conditions that Packet P_j is in the influence scope of Packet P_i :*

$$(1) d_0 = \frac{P_i.a(P_i.y - P_j.y) - P_i.b(P_i.x - P_j.x)}{\sqrt{P_i.a^2 + P_i.b^2}} \leq \frac{D}{2};$$

$$(2) \sqrt{(P_i.x - P_j.x)^2 + (P_i.y - P_j.y)^2} - d_0^2 \leq P_i.d.$$

Set $\text{Space}(i,j) = \text{true}$ if the two packets coincide with the conditions, otherwise $\text{Space}(i,j) = \text{false}$.

Proof. Due to the limit space, the proof is ignored.

Proposition 2. *The maximum value for the initial affecting distance of a vehicle d is the road width D .*

Proof. Due to the limit space, the proof is ignored.

Proposition 3. *If packet P_i stored in Vehicle i and packet P_j stored in Vehicle j both satisfy the following two points:*

$$(1) \text{Space}(i,j) = \text{true};$$

$$(2) \text{The traveling direction of } i \text{ and } j \text{ are the same.}$$

then Vehicle i doesn't forward the received packet P_j , and set $\text{Forward}(i,j) = \text{false}$, otherwise $\text{Forward}(i,j) = \text{true}$.

Proof. Due to the limit space, the proof is ignored.

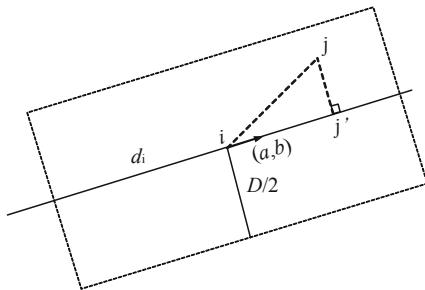


Fig. 3. Influence Scope

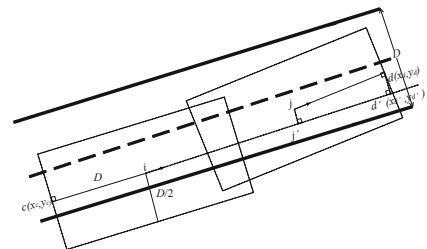


Fig. 4. Data Integration

Integration of Data Packets. In VANETs, urban is the scenario of the network, so streets are the traces of mobile vehicles. This situation has a distinctive characteristic different from other scenario, in which the traffic information in the same road is the same. So after a vehicle stores some traffic information, it can integrate the information in the same road to one piece of information. It can save the local memory.

The format of the integrated packets is the same as that of the broadcast packets, but some variable quantities will change according to the integration. The specific judgements and integration methods will be done as follows.

Proposition 4. If P_i and P_j can be integrated, then the necessary and sufficient conditions are that geographical locations of the two packets have to coincide:

$$(1) d_0 = \frac{P_i.a(P_i.y - P_j.y) - P_i.b(P_i.x - P_j.x)}{\sqrt{P_i.a^2 + P_i.b^2}} \leq \frac{D}{2};$$

$$(2) \sqrt{(P_i.x - P_j.x)^2 + (P_i.y - P_j.y)^2} - d_0^2 \leq L.$$

Set $Road(i,j) = \text{true}$ if the two packets can be integrated, otherwise $Road(i,j) = \text{false}$.

Proof. Due to the limit space, the proof is ignored.

Proposition 5. If packet P_i stored in Vehicle i and packet P_j stored in Vehicle j both satisfy the following four points:

$$(1) Road(i,j) = \text{true};$$

$$(2) \text{The ratio of velocities in } P_i \text{ and } P_j \text{ satisfy that}$$

$$\gamma \leq \frac{P_i.v}{P_j.v} \leq \frac{1}{\gamma}$$

$$(3) \text{The traveling direction of } i \text{ and } j \text{ are the same;}$$

$$(4) \text{Transmission time in } P_i \text{ and } P_j \text{ satisfy that}$$

$$|P_i.t - P_j.t| \leq T$$

Then P_i and P_j can be integrated, and set $Combinable(i,j) = \text{true}$, otherwise $Combinable(i,j) = \text{false}$. Wherein γ is the ordinary velocity ratio constant without any accident.

Proof. Due to the limit space, the proof is ignored.

Proposition 5 is to determine whether the two packets can be integrated. When the data in the two packets satisfy the conditions of Proposition 5, the packets will be integrated. Then they will be combined into one packet (as Fig.4):

Intersect along the extension line of the i and j direction vector to the edge of the influence scopes represented by P_i and P_j respectively to obtain four intersections.

The extension cord equation along the direction of i is $\frac{P_i.y-y}{P_i.x-x} = \frac{P_i.b}{P_i.a}$, namely

$$P_i.a(P_i.y - y) - P_i.b(P_i.x - x) = 0 \quad (2)$$

Simultaneous equations formula (2) and $dist(i,c)$ to get formula (3), and further solve (x_c, y_c) which are the coordinates of c :

$$\begin{cases} P_i.b \times x_c - P_i.a \times y_c + P_i.a \times P_i.y - P_i.b \times P_i.x = 0 \\ \sqrt{(P_i.x - x_c)^2 + (P_i.y - y_c)^2} = D \end{cases} \quad (3)$$

and get two coordinates of c

$$x_c = P_i.x + \frac{P_i.a \times D}{\sqrt{P_i.a^2 + P_i.b^2}}, y_c = P_i.y + \frac{P_i.b \times D}{\sqrt{P_i.a^2 + P_i.b^2}}$$

and

$$x_c = P_i \cdot x - \frac{P_i \cdot a \times D}{\sqrt{P_i \cdot a^2 + P_i \cdot b^2}}, \quad y_c = P_i \cdot y - \frac{P_i \cdot b \times D}{\sqrt{P_i \cdot a^2 + P_i \cdot b^2}}$$

We can get the coordinates of d (x_d, y_d). Mark a perpendicular along i direction vector extension cord over the two d points, whose foot drop is $d'(x_{d'}, y_{d'})$. According to the point to linear distance formula and the Pythagorean Theorem we can get $dist(c, d') = \sqrt{dist(c, d)^2 - (dist(d, d'))^2}$, so

$$dist(c, d') = \sqrt{dist(c, d)^2 - \left(\frac{|P_i \cdot a(y_c - y_d) - P_i \cdot b(x_c - x_d)|}{\sqrt{P_i \cdot a^2 + P_i \cdot b^2}} \right)^2} \quad (4)$$

Integrated affecting distance $P_i \cdot d$ is half of the maximum $dist(c, d')$, and then rounding the outer boundary points; center coordinates of the scope are the coordinates of midpoint between c and d ; the new velocity also change with a corresponding adjustment; t_i is the later time of t_i and t_j .

3.3 Broadcasting and Forwarding Strategy

Packets Broadcasting Algorithm. In VANETs, the distribution of vehicles is uneven, which means some places dense, while some places sparse; and during different period, such as the daytime and night, the vehicle densities are different. In the proposed algorithm, we assume that the distribution is average, and calculate the average vehicle density ρ in the area. In each time T , a vehicle generates a new data packet storing the local traffic information and broadcast by the following method. Vehicle i generates data packet P_i in Δt , deciding whether it will broadcast the packet with the probability q . So each vehicle has a probability $1-q$ not to transmit packet. When it decides to broadcast the packet, it will generate a random number to be delay δ_i , and broadcast the packet when the delay arrives.

Vehicles broadcast packets with probability q , and probability q satisfies $q > 1 - e^{\ln \varepsilon / \rho}$, ε represents threshold value. Set a vehicle send packet with q , we need to ensure a point that there are at least one vehicle send its packet on the same road, in order that the traffic information on this road can be received by the other vehicles. The probability of no vehicle broadcasting its packet is $(1 - q)^\rho$. When the probability is less than a threshold value ε , we can say that the probability of no vehicle whose packet represents the very information broadcasts its packet in the initial time is very small, even almost zero, and such q can be used as a vehicle broadcasting probability. So derive $(1 - q)^\rho < \varepsilon$ can get the choice of q .

Intuitively speaking, the more the number of a vehicle's neighbors, the greater the probability of packets collision sent by vehicles, so the greater the range of delay choice, and the greater the maximum delay. And since the number of a vehicle's neighbors depends on the network density, so we can get $t_d = \alpha\rho$, wherein t_d is the maximum value of delay choice, i.e. the delay random number $\delta_i \in [0, t_d]$. α is a constant factor.

Packets Forwarding Algorithm. In the process of forwarding data, within the communication region of the vehicle, the nearer neighbors of the sending

vehicle receive the data packets more easily than the further neighbors, the probability that they have this very data being larger, so when a vehicle is going to forward its packet, the forwarding probability to the nearer neighbors will be smaller, and to further will be larger; the probability to the vehicles at the edge of the communication region will be the largest. Therefore, when Vehicle i decides to forward packet to its neighbor Vehicle j after processing the data, it will calculate $dist(i,j)$ that the distance from its coordinate to Vehicle j , divide communication radius R to get the forwarding probability:

$$p^F = \frac{dist(i,j)}{R} \quad (5)$$

Vehicles forward the determined packets with the probability p^F , thus reduce the redundancy transmission data.

Algorithm 1. Data Forwarding Algorithm *Deal-Data*

Input: A received packet P_j .

```

1: while Scan data table in Vehicle  $i$  when it isn't to the end do
2:   if  $Forward(i,j) = false$  then
3:     if  $|t_i - t_j| \leq T$  then
4:        $Combine(i,j)$ , forward  $P_i$  with  $p^F$ ,return.
5:     else
6:       if  $t_i < t_j$  then
7:         Replace  $P_i$  with  $P_j$ , forward  $P_j$  with  $p^F$ ,return.
8:       else
9:         Drop  $P_j$ , return.
10:      else
11:        if  $Combinable(i,j) = true$  then
12:           $Combine(i,j)$ , forward  $P_i$  with  $p^F$ ,return.
13:        else
14:          Continue.
15: Store  $P_j$  and forward it with  $p^F$ .

```

4 Simulation Experiments

This section assesses the performance of the strategy proposed by this paper. We will determine some important parameters by simulation, and compare the strategy with other schemes.

4.1 Experimental Environment Settings

In the experimental environment setting, vehicles are traveling within a $3km \times 3km$ fixed region of the street environment. In initialization, each vehicle generates a broadcast period T , broadcast probability q and the maximum delay random number t_d . The simulation of the strategy proposed by this paper executes on NS2[10]. In order to provide a real VANETs environment, the experiment uses SUMO simulator[11] and a street map in Tiger database[12] (Fig.5)

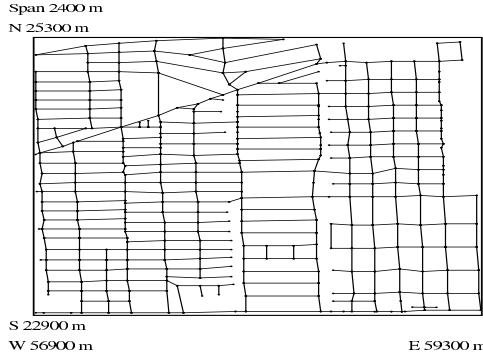


Fig. 5. Simulation Scenario

to generate a street topology model and vehicle moving trajectory file, and the file will be used in NS2.

In the moving scenario there are 2,000 vehicles, which meet the topology of the model, and are limited to a maximum velocity. The size of each data packets are the same, and the local memory can store up to 50 packets.

4.2 The Simulation Comparison Schemes

In the strategy proposed by this paper, there are two key technologies: broadcasting and forwarding strategy. In order to evaluate the performance of the strategy, we will compare this strategy with the other schemes. Since the ordinary broadcast strategy use Epidemic, and there is no broadcast strategy can be used to compare performance, so in this paper we use Epidemic and two other schemes proposed by ourselves to compare with the strategy.

Scheme 1 uses broadcast packets directly on the broadcasting, while uses the algorithms proposed by this paper on the forwarding.

Scheme 2 stores and forwards packets directly on the receiving, while uses the algorithms proposed by this paper on the broadcasting.

4.3 Parameter Settings

First, determine the optimum q and α values by experiments. These two values are important, which determine the efficiency of initial broadcasting.

Setting Probability q . Fig.6 shows the number of initial broadcast packets under predetermined scenario observed by changing the q value, and determine the optimum q value by weighing information area and the number of packets.

In Fig.6 each line represents a α set value. In Fig.6(a), the percentage of the broadcast range is maintained within a relatively large and stabilized range at $q \geq 0.6$; in Fig.6(b), the number of broadcast packets is relatively stable at $0.6 \leq q \leq 0.8$, while it is increasing at $q > 0.8$. So in the initial broadcasting, it is ideal to set the probability within the range from 0.6 to 0.8.

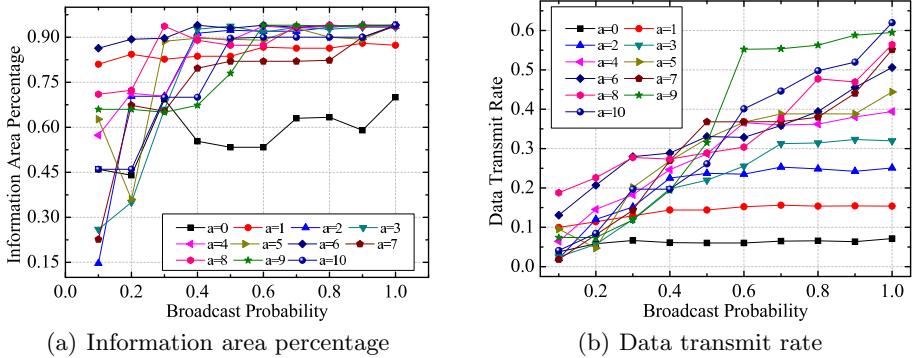


Fig. 6. The affect of q value for initial broadcasting

Setting Maximum Delay α . Fig.7 shows the effect of transmitting data packets under different α value, including the data collision rate and the size of the transmission delay. Weighing the two performances can get a better α value to make the effect of transmitting packets better.

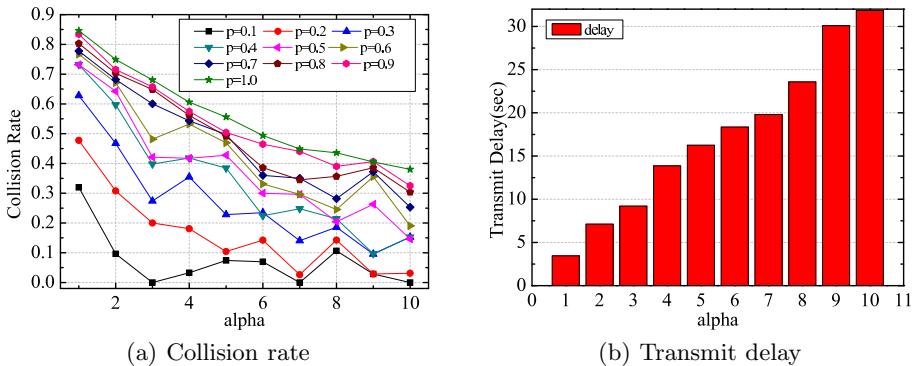
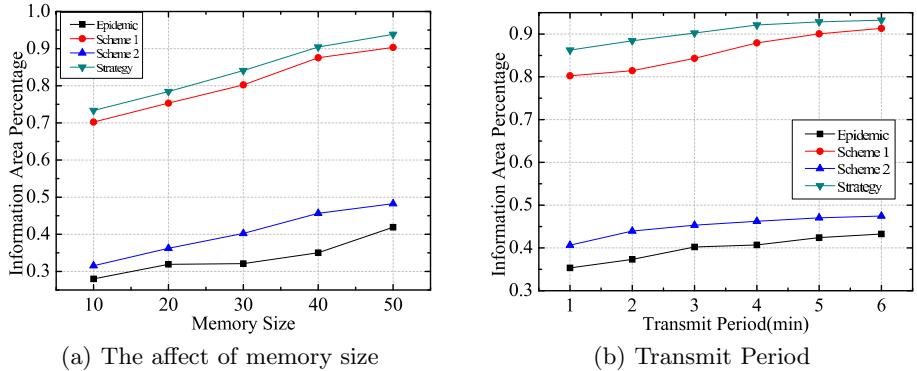
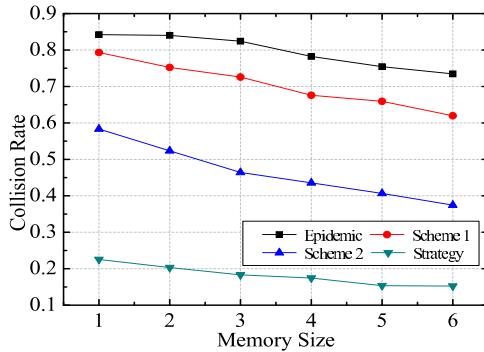


Fig. 7. The affect of α value

In Fig.7 each line represents a probability value. In Fig.7(a) the collision rate stabilizes and maintains in a low state after $\alpha = 6$; according to Fig.7(b), under the premise of maintaining a low collision rate, the delay is smaller choosing α value from 6 to 7.

4.4 Strategy Performance Comparison

Compare the strategy proposed by this paper with schemes mentioned in section 4.2. Wherein q value and α value are determined based on the optimized values obtained in the previous experiment.

**Fig. 8.** Information area percentage**Fig. 9.** Data collision rate

Information Area. Aimed at the issue in the paper, this strategy ultimately wants to make the sum of information area covered by the packets stored in the vehicle memory achieve to the largest. This performance is equivalent to data arrival rate in other strategies. In Fig.9 we compare this strategy with the above-mentioned schemes about information area for some parameters changing.

Fig.8 shows that different methods of processing data in memory results in different effects of information area. Scheme 2 didn't make judgments and processing to the receiving packets, which leads to large redundant data in the memory, and being much less useful data. While Epidemic doesn't take account in this aspect, so the performance in this field is not good neither.

Data Collision Rate. It's necessary to consider data collisions in the broadcasting. By the comparison between this strategy and the other schemes about data collision rate in Fig.9, if vehicles all broadcast without any limit, the data collision rate will be very high; while if all received packets are forwarded, the network congestion will also be large, and so do the data collisions.

5 Conclusion

This paper proposes a data broadcast strategy for traffic information query. This strategy is more than ready for the preparation work for traffic information query. As long as requirements want to collect some data in a specific area directionally and need real-time, they all can broadcast data with this strategy. Vehicles using the strategy can make data be transmitted with greater probabilities, smaller collisions and more comprehensive range, according to store and forward conditions proposed by this paper, so that the data covering information area in vehicles can be as large as possible. Strategies are conducive to real-time traffic data query. Simulation results show that the proposed strategy can achieve the expected performance.

References

1. Wang, L., Wakikawa, R., Kuntz, R., Vuyyuru, R., Zhang, L.: Data naming in Vehicle-to-Vehicle communications. In: Proc. of IEEE, INFOCOM 2012, pp. 328–333 (2012)
2. Hsiao, H.-C., Bai, F.: Flooding-Resilient Broadcast Authentication for VANETs. In: Proc. of ACM, MobiCom 2011 (2011)
3. Lu, N., Luan, T.H., Wang, M., Shen, X., Bai, F.: Capacity and delay analysis for social-proximity urban vehicular networks. In: Proc. of IEEE, INFOCOM 2012, pp. 1476–1484 (2012)
4. Bettstetter, C., Hartenstein, H., Perez-Costa, X.: Stochastic Properties of the Random Waypoint Mobility Model: Epoch Length, Direction Distribution, and Cell Change Rate. In: Proc. of ACM, MSWiM 2002, pp. 7–14 (2002)
5. Pei, G., Gerla, M., Hong, X., Chiang, C.-C.: Wireless Hierarchical Routing Protocol with Group Mobility (WHIRL)
6. Pei, G., Gerla, M., Hong, X.: LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility
7. Zhang, F., He, L., He, W., Liu, X.: Data perturbation with state-dependent noise for participatory sensing. In: Proc. of IEEE, INFOCOM 2012, pp. 2246–2254 (2012)
8. Farnoud (Hassanzadeh), F., Valaee, S.: Reliable Broadcast of Safety Messages in Vehicular Ad Hoc Networks. In: Proc. of IEEE, INFOCOM 2009, pp. 226–234 (2009)
9. Vahdat, A., Becker, V.D.: Epidemic routing for partially connected ad hoc networks. Technique Report, CS-2000-06
10. The Network Simulator, <http://www.isi.edu/nsnam/ns>
11. [EB/OL] (2010), http://sourceforge.net/apps/mediawiki/sumo/index.php?title>Main_Page
12. U. C. Bureau. Tiger, tiger/line and tiger-related products

A Spatial Proximity Based Compression Method for GML Documents

Qingting Wei^{1,2} and Jihong Guan^{1,*}

¹ Dept. of Computer Science and Technology, Tongji University, Shanghai, China

² School of Software, Nanchang University, Nanchang, Jiangxi, China

qtwei2008@gmail.com, jhguan@tongji.edu.cn

Abstract. Geography Markup Language (GML) has become a standard for encoding and exchanging geographic data in various geographic information systems (GIS) applications. Whereas, high-precision spatial data in GML documents often causes high cost in GML storage and transmission as well as parsing. In this paper, we propose a spatial proximity based GML compression method for GML document compression, which transforms spatial data (coordinates in GML documents) into blocks of coordinates and compress coordinates effectively. Concretely, ordered coordinate dictionaries are constructed firstly, and coordinates are encoded as their ordinal numbers in the coordinate dictionaries. Then, delta encoding and LZW encoding are employed to compress the coordinate dictionaries and coordinate ordinal numbers respectively. Finally, the output of the delta encoder and LZW encoder is streamed to a spatial data container. Extensive experiments over real GML documents show that the proposed method outperforms the existing major XML and GML compression methods in compression ratio, while maintaining an acceptable compression efficiency.

Keywords: GML compression, Spatial proximity, Delta encoding.

1 Introduction

As an XML-based geographic modeling language proposed by Open Geospatial Consortium (OGC), Geography Markup Language (GML) has become an international standard [1] for encoding and exchanging geographic data in various Geographic Information System (GIS) applications. Similar to XML documents, GML documents are pretty readable while their redundant structural tags and text-like file storage usually make them extreme large in size. The worse is that GML documents are often much bigger than XML documents for containing a great amount of high-precision spatial data. In GML documents, geometric characteristics of geographic entities are described by spatial data items, each of which consists of a list of coordinates. If each coordinate is stored in binary format, it needs only 4 bytes (by using float type) or 8 bytes (by using double type). However, in GML documents, each coordinate is stored in characters format and its storage space is up to tens of bytes (depending on precision).

* Corresponding author.

The huge size of GML documents causes heavy cost on storage and transmission. As a result, GML compression becomes an important problem in GIS applications, especially in the mobile GIS scenarios where storage and transmission bandwidth are still and will continue to be relatively scarce resources. The existing compression techniques for plain text files and XML documents are applicable to GML compression, nevertheless the compression performance is not satisfactory. On the one hand, plain text compression techniques [2–6] do not distinguish structural tags and data in GML documents. On the other hand, XML compression techniques [7–12] focus only structural tags, without an effective compression on spatial data in GML documents. We developed the first GML specific compressor GPress [13, 14], which compresses structural tags and data in GML documents separately and adopts a floating-point delta encoding scheme in compressing spatial data. The compression principles of GPress are followed by the other GML compressors [15–17]. However, this floating-point delta encoding scheme is not effective and efficient in compressing all coordinates one by one in their occurring order in a GML document. Because coordinates are not always monotonically increasing or decreasing and there may exist repeated X-coordinates, Y-coordinates or Z-coordinates in GML documents.

To compress GML documents effectively and efficiently, in this paper we propose GPress++, a spatial proximity based GML compression method. GPress++ is an improved version of GPress [13, 14], it is more effective in compressing spatial data, by transforming the spatial data items into blocks of coordinates (each block contains a fixed number of coordinates) and compressing the coordinates in each block via the following steps: constructing ordered coordinate dictionaries, encoding coordinates as their ordinal numbers in coordinate dictionaries, compressing coordinate dictionaries by applying delta encoding and compressing coordinate ordinal numbers by applying LZW encoding. We implemented GPress++ and conducted extensive experiments on real GML documents to evaluate the performance of GPress++. Experimental results show that the proposed method outperforms the state of the art XML and GML document compressors in compression ratio, while maintaining an acceptable compression efficiency.

2 The GPress++ Compressor: Architecture and Techniques

Spatial proximity refers to the duplication and nearness of spatial coordinates or points. The proposed GPress++ compressor exploits spatial proximity to improve the compression of spatial data in GML documents. In this section, we present the architecture of GPress++ and the implementation detail on the compression of spatial data.

2.1 The Architecture

Figure 1 show the architecture of GPress++. Similar to GPress, while doing compression, GPress++ parses a GML document firstly. The parsed structural

tags are encoded as integers and saved in the structure container. The parsed data is divided into different groups according to the paths to the root element. Data in different groups is compressed by different semantic compressors and directed to the corresponding data containers. All containers are allocated in memory, the contents of which are compressed by the *gzip* compressors and written to the output compressed GML document finally. Specially, spatial data items are grouped and compressed by the *Coordinate Compressor* and forwarded to the spatial data container.

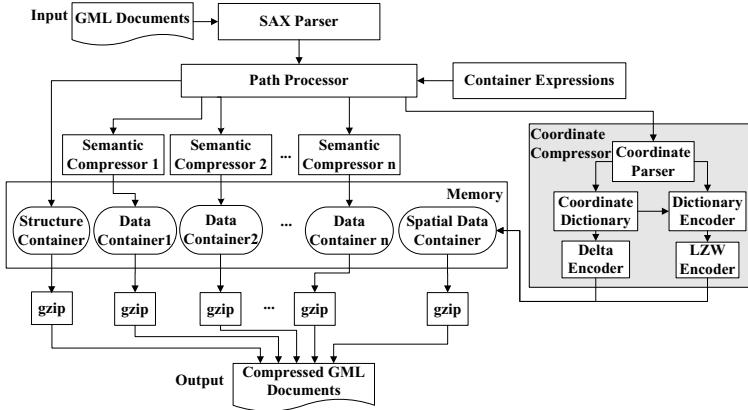


Fig. 1. The architecture of GPress++

As shown in Figure 1, the *Coordinate Compressor* comprises of five major modules, i.e., the *Coordinate Parser*, the *Coordinate Dictionary*, the *Dictionary Encoder*, the *Delta Encoder* and the *LZW Encoder*. In compression, the *Coordinates Parser* module parses spatial data items into blocks of coordinates. When coordinates in a block reach a number, the *Coordinate Dictionary* module is called to collect unique coordinates in the block into ordered coordinate dictionaries. Then the *Dictionary Encoder* module encodes all coordinates in the block as their ordinal numbers at the coordinate dictionaries. Next, the *Delta Encoder* module compresses the ordered coordinate dictionaries and the *LZW Encoder* module compresses the sequence of coordinate ordinal numbers. Finally, the delta encodings and LZW encodings are streamed to the spatial data container. The decompression procedure of the *Coordinate Compressor* is inverse to the compression procedure. Due to limitations of space, we are not going to discuss the decompression procedure in this paper.

2.2 Implementation of the Coordinate Compressor

In this subsection, we present the main implementation detail of the coordinate compressor, including the data structures and the various encoding schemes.

- **Data Structures.** In the first scan of a coordinate block, GPress++ collects unique X-coordinates, Y-coordinates and Z-coordinates by constructing the corresponding coordinate dictionaries where all entries are ordered by coordinate value. Input coordinates are represented by the arrayed BigFloat objects and ordered coordinate dictionaries are implemented by Red-black(RB) trees.

Each BigFloat object consists of the following four members: “isneg” is a character flag with the value 0 or 1 to indicate whether the current coordinate value is negative, 0 for positive and 1 for negative; “point” is an unsigned short integer to save the decimal precision; “value” is a char array to store an integer value transformed from the current coordinate string except for the sign and radix point, where the array capacity is a constant N and the first byte is the lowest byte; “len” is an unsigned short integer to save the size of the “value” array used.

Each internal node in a RB tree stores a tuple (b_i, j) where b_i represents a pointer to the i -th arrayed BigFloat object, and j represents the ordinal number of the pointed BigFloat object in the current RB tree. There does not exist two tree nodes where the pointed BigFloat objects are equal. For any two BigFloat objects b_1 and b_2 : $b_1 < b_2$ iff $b_1.\text{len} < b_2.\text{len}$ or ($b_1.\text{len} = b_2.\text{len}$ and ($\exists i$)($b_1.\text{value}[i] < b_2.\text{value}[i]$ and ($\forall j$)($b_1.\text{value}[i+j] = b_2.\text{value}[i+j]$))), where $i \in [0, N)$, $j \in [1, N - i]$.

- **Encoding Schemes.** In the second scan of a coordinate block, GPress++ puts all coordinates one by one into a sequence, and encodes them as their ordinal numbers in coordinate dictionaries. We let the maximum block size be the product of dimension and 255, so the value of an ordinal number is always less than 255 and one byte is enough to store it.

Moreover, GPress++ applies the floating-point delta encoding [13, 14] to compressing coordinates in ordered coordinate dictionaries one by one. Each coordinate is compressed as one to eight bytes. Since the difference of each coordinate from its preceding in the ordered coordinate dictionary is smaller than that from the other coordinates in the front, the sequential delta encoding of coordinates in such a way is spatially optimal.

In addition, GPress++ regards the sequence of coordinate ordinal numbers as the sequence of points, where every two or three successive numbers are considered as a 2/3 dimensional point having the X-coordinate, the Y-coordinate and/or the Z-coordinate. Then, the LZW encoding [6] is employed to compress the sequence of points, that is, the recurring strings of points are encoded as their assigned codes in bits.

3 Experimental Evaluation

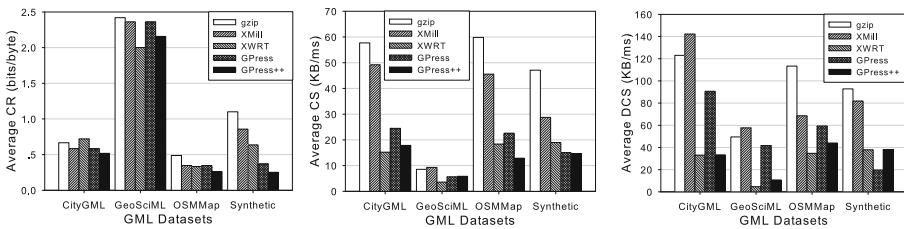
To evaluate the performance of the proposed coordinate compressor, we test 20 GML documents and compare the performance of GPress++ with that of the general text compressor gzip1.2.4 [18], the XML compressors XMill0.7 [7] and XWRT3.2 [12], and the original GPress [13, 14]. The 20 test GML documents

include 15 real documents and 5 synthetic documents. The 15 real documents are selected randomly from three sources, including CityGML [19], GeoSciML [20] and OSMasterMap [21], their sizes ranging from 212KB to 84MB. The five synthetic GML documents are transformed from the same Oracle Spatial sample file about road features of USA and Canada, their size increased from 2.5MB to 1GB. All of the two XML compressors and the two GML compressors (GPress and the proposed GPress++) use the general text compressor gzip at back end. All the compressors are pre-compiled C++ applications which can be executed directly. All experiments are conducted on a PC with a 3.10GHz Intel Core i3-2100 CPU and 3GB RAM, running Windows XP OS.

We evaluate the performance of compressors by the following metrics: 1) **Compression Ratio (CR)** = $\frac{\text{size of compressed document} \times 8}{\text{size of original document}} (\text{bits}/\text{byte})$; 2) **Compression /Decompression Speed (CS/DCS)** = $\frac{\text{size of original document}}{\text{elapsed time}} (\text{KB}/\text{ms})$. A compressor performs better if it achieves lower CR values and is more efficient if it achieves higher CS/DCS values.

Figure 2(a) illustrates the average CR results of the five compared compressors over 20 test documents. The averaged CRs of all compressors lie between 0.2 and 2.5, and GPress++ outperforms the other compressors in averaged CRs over almost all datasets except performing worse than XWRT over the GeoSciML dataset. Especially, on the Synthetic dataset, GPress++ achieves a 77.1% improvement on gzip and a 32.0% improvement on GPress in averaged CR.

Figure 2(b) and Figure 2(c) illustrates the averaged CS and DCS results of the five compared compressors over 20 test documents. The averaged CS values of all compressors lie between 9.3 to 60.0, and the averaged DCS values lie between 4.6 to 142.2. GPress++ performs moderately in averaged CS or DCS however runs faster than XWRT and/or GPress over serval datasets. For example, on the GeoSciML dataset, GPress++ achieves a 63.5% improvement on XWRT in averaged CS. On the Synthetic dataset, GPress++ achieves a 95.2% improvement on GPress in averaged DCS.



(a) Average CR vs. dataset (b) Average CS vs. dataset (c) Average DCS vs. dataset

Fig. 2. Results of compression ratio, compression speed and decompression speed

4 Conclusion

High-precision spatial data in GML documents often causes heavy cost in GML storage and transiting as well as parsing. In this paper, we propose a spatial

proximity based GML compression method GPress++, which transforms spatial data into blocks of coordinates and compress coordinates effectively. Experiments over 20 GML documents show that averagely the GPress++ compressor outperforms major existing compressors, including the general text compressor gzip, the state of the art XML compressors XMill and XWRT, and GML compressor GPress in compression ratio, while maintaining an acceptable compression efficiency. Therefore, GPress++ is especially suitable for the applications such as mobile services, where resources of network and at mobile ends are very pressing. The future work is improving the encoding schemes in GPress++ to optimize interaction with compressed GML data.

Acknowledgments. This research was partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 61173118. Jihong Guan was also supported by the “Shuguang” Program of Shanghai Municipal Education Commission and the Fundamental Research Funds for the Central Universities. Qingting Wei was also supported by the Science and Technology Project of Jiangxi Provincial Department of Education under grant No. GJJ12147.

References

1. Geospatial information – Geography Markup Language (GML). ISO 19136:2007 (2007)
2. Huffman, D.A.: A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE* 40(9), 1098–1101 (1952)
3. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* IT-23(3), 337–343 (1977)
4. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. *Communications of the ACM* 30(6), 520–540 (1987)
5. Cleary, J.G., Witten, I.H.: Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications* 32(4), 396–402 (1984)
6. Welch, T.A.: A Technique for High-Performance Data Compression. *IEEE Computer* 17(6), 8–19 (1984)
7. Hartmut, L., Suciu, D.: XMill: an efficient compressor for XML data. In: *ACM SIGMOD 2000*, pp. 153–164. ACM Press, New York (2000)
8. Cheney, J.: Compressing XML with multiplexed hierarchical PPM models. In: *DCC 2001*, pp. 163–172. IEEE Press, New York (2001)
9. Tolani, P.M., Haritsa, J.R.: XGrind: a query-friendly XML compressor. In: *ICDE 2002*, pp. 225–234. IEEE Press, New York (2002)
10. Min, J., Park, M., Chung, C.: XPress: a queriable compression for XML data. In: *ACM SIGMOD 2003*, pp. 122–133. IEEE Press, New York (2003)
11. League, C., Eng, K.: Type-based compression of XML data. In: *DCC 2007*, pp. 272–282. IEEE Press, New York (2007)
12. Skibiński, P., Grabowski, S., Swacha, J.: Effective asymmetric XML compression. *Software: Practice and Experience* 38(10), 1024–1047 (2008)
13. Guan, J., Zhou, S.: GPress: Towards effective GML documents compression. In: *ICDE 2007*, pp. 1473–1474. IEEE Press, New York (2007)
14. Guan, J., Zhou, S., Chen, Y.: An effective GML documents compressor. *IEICE Transactions on Information and Systems* E91-D(7), 1982–1990 (2008)

15. Wei, Q., Guan, J.: A GML Compression Approach Based on On-line Semantic Clustering. In: Geoinformatics 2010, pp. 1–7. IEEE Press, New York (2010)
16. Wei, Q.: A Query-Friendly Compression for GML Documents. In: Xu, J., Yu, G., Zhou, S., Unland, R. (eds.) DASFAA Workshops 2011. LNCS, vol. 6637, pp. 77–88. Springer, Heidelberg (2011)
17. Yu, Y., Li, Y., Zhou, S.: A GML Documents Stream Compressor. In: Xu, J., Yu, G., Zhou, S., Unland, R. (eds.) DASFAA Workshops 2011. LNCS, vol. 6637, pp. 65–76. Springer, Heidelberg (2011)
18. GZip 1.2.4., <http://www.gzip.org>
19. Kolbe, T.H.: CityGML - Exchange and Storage of Virutual 3D City Models (2002), <http://www.citygml.org>
20. CGI Interoperability Working Group: The GeoSciML project (2003), <http://www.geosciml.org/>
21. Ordnance Survey: OS MasterMap (2001), <http://www.ordnancesurvey.co.uk>

Efficient MSubtree Results Computation for XML Keyword Queries

Junfeng Zhou, Jinjia Zhao, Bo Wang, Xingmin Zhao, and Ziyang Chen

School of Information Science and Engineering, Yanshan University, Qinhuangdao, China
`{zhoujf, jinjia, bowang, xingmin, zychen}@ysu.edu.cn`

Abstract. In this paper, we focus on efficient constructing of *matched subtree* (*MSubtree*) results for keyword queries on XML data. We firstly show that the performance bottlenecks for existing methods lie in (1) computing the set of *relevant keyword nodes* (RKNs) for each subtree root node, (2) constructing the corresponding MSubtrees, and (3) parallel execution. We then propose a two-step *generic top-down* subtree construction algorithm, which computes SLCA/ELCA nodes in the first step, and *parallelly* gets RKNs and generates MSubtree results in the second step, where *generic* means that (1) our method can be used to compute different kinds of subtree results, (2) our method is independent of the query semantics; *top-down* means that our method constructs each MSubtree by visiting nodes of the subtree constructed based on an RKN set *level-by-level* from *left to right*, such that to avoid visiting as many useless nodes as possible.

1 Introduction

An XML document is typically modeled as a node-labeled tree T . For a given keyword query Q , each result T_v is a subtree of T containing each keyword of Q at least once, where the root node v of T_v should satisfy a certain query semantics, such as SLCA [4] or ELCA [1]. Based on the set of qualified root nodes, researchers proposed three kinds of subtrees: (1) *complete subtree* (*CSubtree*) T_v^C , which is rooted at a node v and is excerpted from the original XML tree without pruning any information; (2) *path subtree* (*PSubtree*) T_v^P , which consists of paths from v to all its descendants, each of which contains at least one input query keyword; (3) *matched subtree* (*MSubtree*) T_v^M [3], which is a subtree after removing redundant information from T_v^P based on specific constraints. Let S_u be the set of distinct query keywords contained in the subtree rooted at node u , $S_u \subseteq Q$. Intuitively, a subtree T_v is an MSubtree if for any descendant node u of v , there does not exist a sibling node w of u , such that $S_u \subset S_w$, which we call as the constraint of “*keywords subsumption*”.

Example 1. For query Q_1 and XML document D_1 in Fig. 1, the qualified SLCA node is node 7, and ELCA nodes are node 2 and 7. If the adopted query semantics is SLCA, then the *CSubtree* T_7^C is the subtree rooted at node 7, the *PSubtree* T_7^P and *MSubtree* T_7^M are shown in Fig. 1. If the query semantics is ELCA, then the *CSubtrees* are the two subtrees rooted at node 2 and 7; the *PSubtrees* are T_2^P and T_7^P ; and the *MSubtrees* are T_2^M and T_7^M , respectively. \square

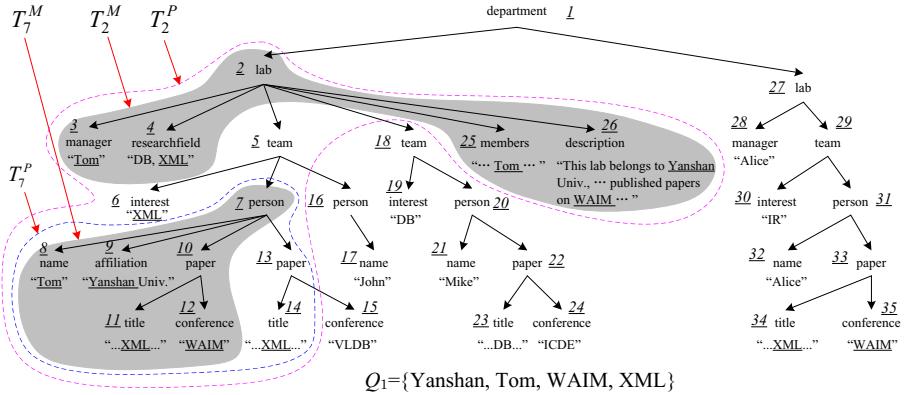


Fig. 1. A sample XML document D_1 . The number beside each node v is its ID value, which is compatible with the document order, i.e., the depth-first left-to-right visiting order.

From Example 1 we know that *CSubtree* and *PSubtree* could make users feel frustrated since they may contain too much irrelevant information. As a comparison, an MSubtree contains all necessary information after removing nodes that do not satisfy the constraint of keywords subsumption, and is more self-explanatory and compact.

However, most existing methods [4, 6] addressing efficiency focus on computing qualified root nodes, such as SLCA or ELCA nodes, as efficient as possible. In fact, constructing MSubtrees is not a trivial task. [3] adopts a *three-step* strategy to get all MSubtrees: **(Step1)** scans all nodes in the set of inverted lists to get SLCA results. **(Step2)** gets a set of *relevant keyword nodes* (RKNs) for each SLCA result v by rescanning *all* these nodes again, this RKN set covers, in the set of inverted lists, all v 's descendant nodes that are not contained by other subtrees rooted at v 's descendant LCA nodes. **(Step3)** constructs a PSubtree for each RKN set, and then apply the constraint of keywords subsumption on this PSubtree to get the corresponding MSubtree. The problems of [3] lie in two aspects: (1.1) no matter how many useless nodes can be skipped in Step1, Step2 will visit all these nodes to get all RKN sets; (1.2) no matter how many useless nodes are contained by each PSubtree, they all need to firstly construct the PSubtree containing these useless nodes, and then make pruning.

Further, the wide availability of commodity multi-core (processor) systems allows concurrent execution of multiple threads through the use of multiple simple processing units (cores) with shared-address space on a single chip. For keyword query processing on XML data, however, prior algorithms assume a uniprocessor architecture, ignoring the potential for performance improvement that a chip multi-core system can offer. With the increase of the number of cores (processors) on a single chip, it is imperative to incorporate this hardware trends in algorithm design, for which the core problem is designing a good (if not best) data partitioning technique, such that different threads can be executed independently.

In this paper, we focus on efficiently computing all MSubtrees rooted at SLCA or ELCA nodes on a multi-core system. We firstly propose a *generic top-down* subtree construction strategy, where “*generic*” means that our method is independent of the

pos	1	2	1	2	3	1	2	3	1	2	3	4	5			
L_1	1	1	1	2	2	1	2	27	1	1	1	1	1	2	27	29
	2	2	26	3	5	26	5	25	2	2	31	4	6	7	7	31
	5	7		7	8		10	33	12	35		10	13	33	11	14
	7	9														34

Fig. 2. Inverted lists of “Yanshan” (L_1), “Tom” (L_2), “WAIM” (L_3) and “XML” (L_4), where “pos” denotes the array subscript of each Dewey label in an inverted list

query semantics, i.e., SLCA or ELCA; “*top-down*” means that our method constructs each MSubtree by visiting nodes of the corresponding PSubtree *level-by-level* from *left to right*, such that to avoid visiting as many useless nodes as possible. Based on the *top-down* subtree construction strategy, we propose a two-step algorithm, namely TDM, for XML keyword query processing on XML data. Same as [3], Step1 of our method is used to find qualified SLCA/ELCA nodes. Different with [3], Step2 of our method is used to compute all MSubtrees, that is, Step2 is the union of the last two steps of [3]. We show that based on our top-down subtree construction strategy, for all SLCA/ELCA nodes, Step2 of our method can be executed independently by different threads. As Step1 usually consumes less than 10% overall processing cost, the parallel processing of Step2 significantly improves the overall performance.

2 Preliminaries

We model an XML document as a node-labeled tree, where nodes represent elements or attributes, while edges represent direct nesting relationship between nodes. If a keyword k appears in the node name, attribute name, or text value of v , we say v *directly contains* k ; otherwise, if k appears in the subtree rooted at v , we say v *contains* k . In Fig. 1, we assign each node an ID, which is its depth-first visiting order, based on which we can get the variant of its Dewey label by concatenating all IDs of its ancestors. E.g., for node 7, we denote its Dewey label as “1.2.5.7”. Given two nodes u and v , $u \prec_a v$ denotes u is an ancestor node of v , $u \preceq_a v$ denotes $u \prec_a v$ or $u = v$.

For a given query $Q = \{k_1, k_2, \dots, k_m\}$ and an XML document D , we use L_i to denote the inverted Dewey label list of k_i . Fig. 2 shows the inverted lists of Q_1 . Among existing query semantics, Smallest Lowest Common Ancestor (SLCA) [4] defines a subset of all Lowest Common Ancestors (LCAs) of Q , of which no LCA is the ancestor of any other LCA. In Fig. 1, the qualified SLCA node of Q_1 on D_1 is node 7. Compared with SLCA, ELCA tries to capture more meaningful results. A node v is an ELCA if the subtree T_v rooted at v contains at least one occurrence of all query keywords, after excluding the occurrences of the keywords in each subtree $T_{v'}$ rooted at v 's descendant LCA node v' . In Fig. 1, the ELCAs of Q_1 are node 2 and 7.

Given a query Q , assume that v is an x LCA node (x LCA can be either SLCA or ELCA), and keyword node u satisfy $v \preceq_a u$, we say u is a *relevant keyword node* (RKN) of v if there does not exist other LCA nodes of Q on the path from v to u . The

set of RKNs of v is denoted as $RKN(v)$, which contains at least one keyword node w.r.t. each keyword of Q . E.g., for query Q_1 and D_1 in Fig. 2, ELCA nodes are node 2 and 7, $RKN(2) = \{3, 4, 25, 26\}$, $RKN(7) = \{8, 9, 11, 12, 14\}$. Note that since node 5 is an LCA node, according to RKN's definition, node 6 is not an RKN of node 2, because node 6 is not taken into account when checking node 2's satisfiability.

3 The TDM Algorithm

Our method adopts a two-step strategy to get all MSubtrees. In the first step, it gets a set of x LCA nodes by calling either one of the state-of-the-art algorithms [1, 4, 6]. In the second step, it firstly computes the RKN set for each x LCA node v , then constructs the corresponding MSubtree. “TDM” means that our method computes all MSubtrees in a *top-down* way.

3.1 Computing RKN Sets

To get all RKNs for each x LCA node v , our method directly takes all descendant nodes of v in each inverted list as RKNs of v , which we call as the *relaxed* interval sets of v . In this way, the task of computing RKNs for all x LCA nodes can be executed *in parallel*. Although such interval sets may contain *non-RKNs* for each x LCA node v , our algorithm can wisely avoid processing those useless nodes when constructing MSubtrees.

3.2 MSubtree Construction

The main idea is: *adding nodes to T_v^M in a top-down way, whenever it finds a useless node u , all other nodes in T_u^P will be directly pruned*. The technical problems we need to solve are: ($P1$) how to know, for a given node u , the keyword set S_u without visiting its descendant keyword nodes; ($P2$) as many sibling nodes may correspond to the same keyword set, and all these nodes may be valid nodes of an MSubtree, we need to know how to reduce the cost of checking whether the given keyword set S_u is subsumed by, or subsumes that of u 's sibling nodes.

For $P1$, the checking of whether a node u in the j^{th} level of the given XML tree contains k_i is very simple: *if u appears in the j^{th} level of L_i , then u contains k_i* . Given the current node u , the problem is how to get the next distinct node ID. This problem can be efficiently solved by using binary search, which takes u 's ID as the key to probe all IDs of the same level of L_i to find the *smallest* ID that is greater than u 's ID.

For $P2$, we proved that the number of distinct keyword sets that are not subsumed by, or subsume other keyword sets is bounded by $C_m^{\lfloor m/2 \rfloor}$. Based on this result, we distribute all u 's child nodes into different distinct keyword sets that are not subsumed by, or subsume other keyword sets. In this way, the operation of checking the subsumption relationship of keyword sets for different sibling nodes can be transformed into directly checking the subsumption relationship between different keyword sets. Compared with MaxMatch, our method reduces the cost of checking the subsumption relationship for each node from $O(2^m)$ to $O(C_m^{\lfloor \frac{m}{2} \rfloor})$.

Example 2. To construct T_7^M for Q_1 , our method firstly processes node 7, then adds node 8, 9 and 10 to three node sets $V_{\{Tom\}}$, $V_{\{Yanshan\}}$ and $V_{\{XML, WAIM\}}$, then adds these node sets into a temporary queue \mathcal{Q} one by one. The next processed node is node 13, as $S_{13} = \{\text{XML}\} \subset \{\text{XML, WAIM}\}$, node 13 will be skipped. After that, we remove node 8, 9 and 10 from \mathcal{Q} and insert them into T_7^M , then we delete $V_{\{Tom\}}$, $V_{\{Yanshan\}}$ and $V_{\{XML, WAIM\}}$ from \mathcal{Q} . The last two nodes that will be processed are child nodes of node 10, i.e., node 11 and 12, which are added into two node sets and inserted into \mathcal{Q} . After that, the two nodes are removed from \mathcal{Q} and inserted into T_7^M . \square

3.3 Execution on a Multicore System

We consider two kinds of strategies for executing Step2 of our method in parallel on a multicore system: (S1) each thread actively steals an x LCA node from the set of input x LCA nodes after it completes constructing the previous MSubtree; (S2) actively distribute all x LCA nodes evenly to all threads.

The benefit of the first strategy lies in that if different MSubtrees have huge difference in their sizes, it can roughly guarantee that different threads have similar workload; however, when the size of all MSubtrees is small, it may suffer from more communication and synchronization cost between these threads. As a comparison, the benefit of the second strategy lies in that it can completely avoid the communication and synchronization cost between different threads, but when different MSubtrees have huge difference in their sizes, it cannot guarantee that different threads have similar workload.

Note that our TDM algorithm can be easily extended to support different query semantics, such SLCA and ELCA, and different kinds of subtree results, such as PSubtree, RTF [2], TMSubtree [5], etc.

4 Experimental Evaluation

Our experiments were implemented on a PC with dual six-core Intel Xeon E5-2620 CPUs running at 2.00GHz, 16GB memory, 500 GB IDE hard disk, and Windows XP professional as the operating system. The algorithms used for comparison include Max-Match [3] and TDM, and were implemented to support both SLCA and ELCA semantics. For ELCA (SLCA) semantics, the algorithm used in Step1 is FwdELCA (Fwd-SLCA) [6]. All these algorithms were implemented using Microsoft VC++. The running time is the averaged over 100 runs with hot cache. We selected 12 queries from XMark (<http://monetdb.cwi.nl/xml>) dataset of 582MB for our experiments.

Fig. 3 (A) shows comparison of the running time, from which we know that TDM performs faster than MaxMatch by at most two orders of magnitude (QX4). The reason lies in that MaxMatch needs to visit all nodes in the set of inverted lists to get all RKNs, and visit all nodes of each T_v^P . Fig. 3 (B) shows the comparison of the running time for the TDM-E algorithm executed by different number of threads on a multicore system, where the number before “T” denotes the number of threads, “S1 (S2)” denotes the *first (second)* parallel strategy (see Section 3.3), “Avg” is the averaged running time for QX1 to QX12. From Fig. 3 (B) we know that the performance gain is not significant when the number of threads becomes large, this is because, with the increase of the

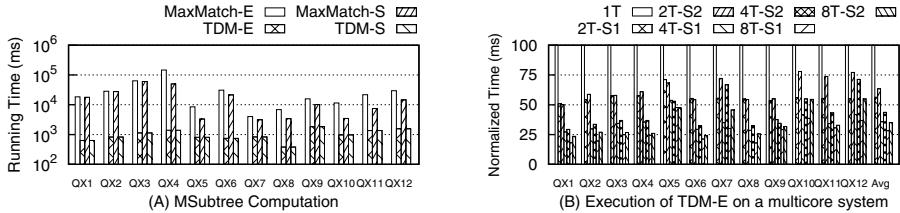


Fig. 3. Comparison of running time

number of threads, the impacts of Step1 is more significant since it cannot be executed in parallel. The third observation is that, since S1 roughly guarantees all threads have similar workload, S1 works better than S2 by 9%, 8% and 6% for two, four and eight threads on average, however, when the whole task is evenly distributed to all threads, S2 is better than S1 by avoiding the synchronization cost (QX5).

5 Conclusions

Considering that existing methods suffer from inefficiency in MSubtree construction, we proposed a two-step *generic top-down* subtree construction algorithm, namely TDM, to support different kinds of query semantics and subtree results, and can be executed in parallel to utilize computational power of a multicore system. The experimental results verified the performance benefits of our method according to various evaluation metrics.

Acknowledgment. This research was partially supported by the National Natural Science Foundation of China (No. 61073060, 61272124), the Research Funds from Education Department of Hebei Province (No. Y2012014), and the Science and Technology research and development program of Hebei Province (No. 11213578).

References

1. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: Ranked keyword search over xml documents. In: SIGMOD Conference (2003)
2. Kong, L., Gilleron, R., Lemay, A.: Retrieving meaningful relaxed tightest fragments for xml keyword search. In: EDBT, pp. 815–826 (2009)
3. Liu, Z., Chen, Y.: Reasoning and identifying relevant matches for xml keyword search. PVLDB 1(1), 921–932 (2008)
4. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: SIGMOD Conference (2005)
5. Zhou, J., Bao, Z., Chen, Z., Ling, T.W.: Fast result enumeration for keyword queries on XML data. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 95–109. Springer, Heidelberg (2012)
6. Zhou, J., Bao, Z., Wang, W., Ling, T.W., Chen, Z., Lin, X., Guo, J.: Fast slca and elca computation for xmlkeyword queries based on set intersection. In: ICDE (2012)

Measuring and Visualizing Interest Similarity between Microblog Users

Jiayu Tang, Zhiyuan Liu, and Maosong Sun

State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, China
{tjy430,lzy}@gmail.com, sms@tsinghua.edu.cn

Abstract. Microblog users share their life status and opinions via microposts, which usually reflect their interests. Measuring interest similarity between microblog users has thus received increasing attention from both academia and industry. In this paper, we design a novel framework for measuring and visualizing user interest similarity. The framework consists of four components: (1) Interest representation. We extract keywords from microposts to represent user interests. (2) Interest similarity computation. Based on the interest keywords, we design a ranking framework for measuring the interest similarity. (3) Interest similarity visualization. We propose a integrated word cloud scenario to provide a novel visual representation of user interest similarity. (4) Annotation data collection. We design an interactive game for microblog users to collect user annotations, which are used as training dataset for our similarity measuring method. We carry out experiments on Sina Weibo, the largest microblogging service in China, and get encouraging results.

Keywords: interest similarity, information visualization, microblogging, keyword extraction.

1 Introduction

Microblogging is a new form of blogging in the Web 2.0 era. Typical microblogging services include Twitter¹ and Sina Weibo². Microblogging services allow users to post small elements of content such as short text messages, images, videos, etc., the so-called *microposts*. Microposts are made by succinctly broadcasting information within a certain length, e.g., 140 English or Chinese characters. Users usually share information, update daily activities, and seek knowledge on microblogging services. Microposts can thus reflect user interests to a certain extent, and we can identify a user's major interests by extracting representative words and phrases in the microposts [1]. We call these words and phrases as *keywords*.

¹ <http://twitter.com/>

² <http://weibo.com/>

As a typical social media, the relations between users in microblogging have attracted many attentions from both research and commercial communities. On a microblogging service, a user may follow and pay close attention to anyone who s/he is interested in. The reasons behind the following behaviors are complicated. For example, a user may follow another one just because they have social connections in the real-world, or because they share similar interests so that they can keep an eye and communicate with each other. Currently, most researches focus on studying the following behaviors through the following-network structure. In fact, however, since the microposts imply the interests of users, we can model user relations via their similarities of interests. Therefore, as an alternative to the perspective of network structure analysis, we use keywords as the representation of user interests, and propose a novel framework to measure the interest similarity between users.

In this paper, a novel framework is proposed for measuring and visualizing interest similarity between microblog users. We extract keywords from microposts to represent user interests. With interest keywords, a support vector machine for ranking (SVM-rank) model is learned, which measures the interest similarity effectively. Then, we extend Wordle [2], a widely-used text visualization, into a integrated word cloud scenario to make viewers comprehend the interest similarity between microblog users clearly and intuitively. Besides, we design an interactive game for microblog users to collect user annotations for SVM-rank model training.

The rest of this paper is organized as follows. We first briefly discuss related work in Section 2, followed by the details of our framework in Section 3. Then, we evaluated and verified our framework on a real-world microblogging service to show the effectiveness of our framework in Section 4. Finally, we conclude with a discussion and future directions in Section 5.

2 Related Work

2.1 Microblogging Analysis

Over the last couple of years, microblogging has been investigated from several perspectives. Java et al. [3] studied the topological and geographical properties of Twitter’s social network to understand the community structure in microblogging. Kwak et al. [4], Wu et al. [5], and Bakshy et al. [6] investigated the diffusion of information on Twitter. Zhao and Rosson [7] qualitatively investigated the motivation of the users who use Twitter. Krishnamurthy et al. [8] analyzed the characteristics of Twitter users in such aspects as classes, behaviors, and social networks.

There are also some researches investigating user interests in microblogging. Using natural language processing tools, Piao and Whittle [9] extracted named entities and core terms to identify individual Twitter users’ interests. Using TFIDF and TextRank, Wu et al. [10] also extracted keywords from the Twitter microposts to label user’s interests and concerns. Yamaguchi et al. [11] extracted tags from the names of “Twitter List” (i.e., user groups) to discover appropriate

topics for list members and identify their common interests. Michelson and Macskassy [12] leveraged Wikipedia as a knowledge base to categorize the entities in the Twitter microposts and built a topic profile for each Twitter user. Banerjee et al. [1] analyzed real time user interests in different cities by mining content-inductive and usage-inductive keywords, each of which represents different class of user interests.

As shown in the above-mentioned works, keywords have been proved to be an appropriate and encouraging way to identify user interests. In this paper, we also aim to extract keywords to estimate microblog users' interests for further measurement.

2.2 Word Cloud

A word cloud, also known as a tag cloud, is a popular way of representing text data. It's typically used in the Web 2.0 era to visualize the frequency distribution of key terms which depict the website content. A word cloud is originally organized in horizontal lines, and popularized in the photo sharing site Flickr³ to show the user-generated tags of photos in 2004. A cloud encodes word importance or frequency information via font size. There have been several tools for generating word clouds from text provided by the Web users. Wordle⁴ is one of the most outstanding and appealing tools. It is widely used in the social community and mainstream media [2]. The word cloud created by Wordle is significantly different from regular ones because of its striking graphic statements: the words are arranged tightly and the display space is thus efficiently used; words can be placed in different orientation.

While Wordle is a state-of-the-art tool for generating a simple word cloud, it cannot show any relationship among words; that is, text content about two or more subjects cannot be recognized by the word cloud created by Wordle. Paulovich et al. [13] and Cui et al. [14] used a sequence of word clouds to show different document collections. Nevertheless, to the best of our knowledge, it is still difficult for current word cloud schemes to demonstrate the commonness and difference between two collections in an intuitive way.

The impact of the visualization on the user experience has also been studied [15]. A comparative study of several word cloud layouts suggested that appropriate layouts should be carefully selected according to the expected user goals [16].

3 Framework

In this section, we describe the framework for measuring and visualizing the similarity between microblog users (shown in Figure 1). We first show how to extract keywords from microposts to represent user interests (3.1). Further, we present a ranking approach to measuring the interest similarity between users

³ <http://www.flickr.com/>

⁴ <http://www.wordle.net/>

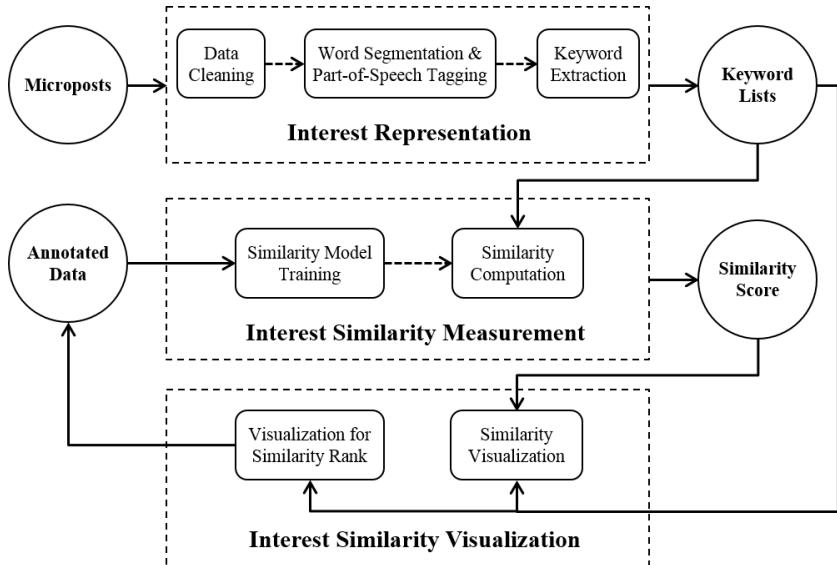


Fig. 1. Framework for measuring and visualizing the interest similarity between microblog users

(3.2). Then, we propose a integrated word cloud visualization to provide a novel and clear representation of user interest similarity (3.3).

3.1 Interest Representation

We identify a microblog user's interests by extracting keywords from his/her microposts. In this paper, we take Sina Weibo, the largest microblogging service in China, as our research service, and get users' microposts from its APIs⁵. On Sina Weibo, an overwhelming majority of the microposts are in Chinese. Hence, we perform Chinese word segmentation and part-of-speech (POS) tagging before keyword extraction.

Data Cleaning. Before word segmentation and POS tagging, we clean microposts and only keep plain text data in preprocessing. In China, a significantly large percentage of microposts are retweets [17]. A retweet usually contains two parts: the original micropost and a comment by the retweeting user. A user may retweet a micropost simply because it's popular, and the micropost usually contains more information about the user who post it originally than that about the retweeter. Therefore, we only retain the comment by the retweeter in a retweet micropost.

A user may use emoticons in microposts to help express his/her sentiment. As emoticons cannot directly help keyword analysis and further similarity

⁵ <http://open.weibo.com/wiki/>

computation, we remove them from the microposts. Additionally, we also remove the user names mentioned in posts, URLs and other none-texts from microposts.

Word Segmentation and POS Tagging. With the clean data, we perform word segmentation and POS tagging using a practical system THULAC [18]. Other POS taggers such as Stanford Log-linear Part-Of-Speech Tagger [19] or Apache OpenNLP POS Tagger⁶ can also be easily embedded in our framework to support other languages. After filtering the stop words, only notional words with specific concepts are selected as candidate words for keyword extraction: common nouns, person names, place names, institute names, idioms and other proper nouns.

Keyword Extraction. With filtered notional words, we perform keyword extraction derived from an efficient and effective framework proposed by Liu et al. [20], in which a translation-based method and a frequency-based method are combined. The translation-based method can summarize appropriate keywords in spite of the noise caused by varied subjects and the frequency-based method can find new words used in microposts. Their experiments on Sina Weibo have shown that this framework is effective and efficient for identifying user interests.

To measure interest similarity, we need to extract keywords from two microblog users' microposts. After keyword extraction, we get two keyword lists, each of which indicates a user's interests. In the keyword list, each keyword is assigned a weight.

3.2 Interest Similarity Measuring

Although the task of computing interest similarity is important for both academia and industry, there is actually no gold standard for directly evaluating the computation of interest similarity between microblog users. It is intuitive to use the cosine similarity of two keyword lists for similarity computation. The cosine scores in isolation, however, are usually too small due to the sparsity of keyword matrix, which cannot well illustrate the similarity between microblog users. In this paper, we propose to take multiple features together for measuring user similarity. For this method, the challenge is how to collect annotation data and design an approach to supervised learning.

We first introduce the method for similarity computation, and then introduce the collection method of annotation data.

Similarity Computation. With two keyword lists, we first use the vector space model to represent keyword lists and compute the cosine similarity between them. We select the cosine score, the number of common keywords and ratios of the number of common keywords to the numbers of two keyword lists as the features for measuring interest similarity between microblog users, using our similarity model learned from a Ranking SVM algorithm [21] (experiments

⁶ <http://opennlp.apache.org/>



Fig. 2. Interface of the game application for collecting annotation data. A user press one Select button to choose whether user *A* or user *B* is more similar to himself/herself.

for the model selection are shown in Section 4.1). The value v derived from the similarity model is mapped into a specific score s by the following sigmoid function:

$$s = \frac{100}{1 + e^{-v}} \quad (1)$$

The final similarity score s is called *interest exponent* with range from 0 to 100.

Annotation Data Collection. To train the SVM-rank model, we design an interactive game for microblog users to collect annotation data. Compared to tell the preference between two users according to interest similarity, it is difficult for an annotator to give an absolute score of his/her similarity with a user. Hence, we ask a microblog user to provide his/her interest preference between his/her two friends. Moreover, if we only show interest keyword lists during the annotation, an annotator is difficult tell exactly how similar two users are. Therefore, in the interactive game, we use our novel visualization method to demonstrate the interest similarities (see Section 3.3 for detailed introduction) to help annotation. We finally implement a Web application to gather annotation data.

Figure 2 shows a snapshot of the game application. With a user ID on Sina Weibo, the application can access user data to generate a pair of word clouds as well as the information about the keyword numbers at each time. A user can click the Select buttons to annotate whether the left two microblog users are more similar or the right two ones, without knowing in advance who they are respectively. Then, the application will show the real identities of user *A* and user *B*. Moreover, the user can share the visualization with his/her annotation as a micropost on Sina Weibo, which can attract more users to use our game application.

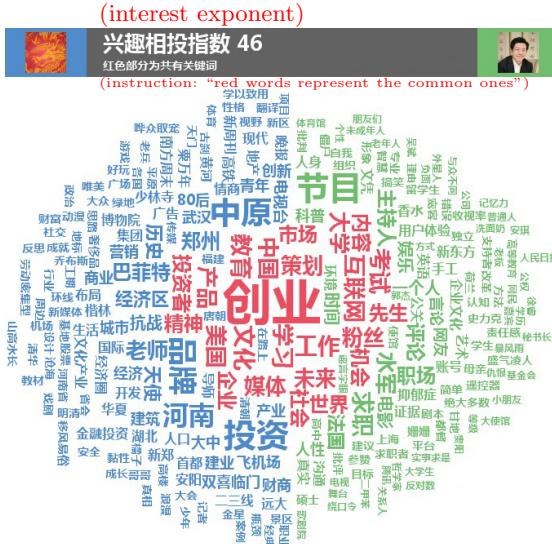


Fig. 3. Interest similarity visualization of two users on Sina Weibo

3.3 Interest Similarity Visualization

With interest keyword lists and their similarity score, we can create similarity visualization for viewers. By showing two microblog users' keywords and their common ones at the same time on canvas as well as the interest exponent, an overall view of interest similarity between two users is provided. Figure 3 shows an example of our similarity visualization, in which the common keywords of two microblog users, such as “创业 (startup),” “互联网 (internet),” and “教育 (education),” are shown in the center of the canvas. Moreover, the avatar of each user is placed on the corresponding side at the top of the canvas to make our visualization more self-explanation for viewers. In this section, we describe the visual representation and the rationale of our visualization.

Word Colors. Different colors are used to represent three kinds of keywords to make the visualization intuitionistic and aesthetic. Assuming that viewers are more interested in the common keywords of two microblog users, these ones are colored in striking red and are placed in the center of the canvas. The distinctive keywords of each user are separately colored in visually distinguishable blue and green.

Font Size. Following the most accepted visual design, font size is used to indicate word weight. Big words catch viewers' attention more easily than small ones [22], and the font size of the common keywords is thus bigger than that of the distinctive ones owned by each user. To balance the distinction and legibility, the variation of the font size of the different keywords has to be critically chosen. The font size $csize_i$ for a common keyword i is calculated as follows:

$$tsize_i = C_{max} - (C_{max} - C_{min}) \sqrt{\frac{w_{max} - w_{H_i}}{w_{max} - w_{min}}} \quad (2)$$

$$csize_i = (\alpha v + \beta) tsize_i \quad (3)$$

where w_{max} is set to the maximum weight of the common keywords, w_{min} is set to the minimum weight of the common keywords, and w_{H_i} is the harmonic mean of the weights of i in two users' keyword lists. C_{max} , C_{min} , α , and β are the constant factors. v is the similarity value got from the similarity model, and the size of the common keyword is thus adjusted according to the similarity between microblog users. The more similar two microblog users are, bigger the size of the common keywords is.

The font size $dsize_j$ for a distinctive keyword j owned by only one microblog user u is calculated as follows:

$$dsize_j = D_{min} + (D_{max} - D_{min}) \left(\frac{w_j}{w'_{max} - w'_{min}} \right)^\mu \quad (4)$$

where w'_{max} is set to the maximum weight of the keywords of u , w'_{min} is set to the minimum weight of the keywords of u , and w_j is the weight of j . D_{max} , D_{min} , and μ are the constant factors.

Word Layout. Since Wordle’s layouts are proven to be very compelling [2] and circular layout with decreasing weight is suitable for finding major concerns [16], we mimic Wordle’s design rationale: the words with high weight are placed centrally on canvas, and the small ones fill the rest spaces to provide a holistic view. The layout of keywords proceeds on the basis of the pseudo code proposed in [2]. The common keywords have high priority when determining where to be placed. The distinctive keywords of each user are placed on one side of the dividing line (shown in Figure 4), which visually separates two users’ keywords.

The number of keywords may vary widely with respect of the amount of users' microposts. To form an aesthetic view, the dividing line of two users' distinctive keywords thus has to be dynamic rather than always in the middle of the canvas.



Fig. 4. A case when the numbers of two users' keywords are quite different

Given two microblog users A and B , user A 's keywords have to be placed on the left side of the dividing line, the boundary of which is $xpos_l$, and user B 's have to be placed on the right side of the dividing line, the boundary of which is $xpos_r$. When placing user A 's keywords, the position of $xpos_l$ is calculated as follows:

$$xpos_l = \frac{(\frac{n_A}{n_B})^\gamma}{1 + (\frac{n_A}{n_B})^\gamma} \quad (5)$$

where n_A is the number of user A 's keywords, and n_B is the number of B 's. γ is the constant factor and we experimentally set it to a value of 0.25. When all of user A 's distinctive keywords are placed on the canvas and user B 's are to be placed, the position of $xpos_r$ is calculated as follows:

$$xpos_r = \min(xpos_l, xpos_{A_{max}}) \quad (6)$$

where $xpos_{A_{max}}$ is set to the rightmost position of user A 's distinctive keywords (shown in Figure 4). Equation 6 thus creates a harmonious visual effects when the numbers and weights of the keywords of two users are quite different.

4 Experiments

4.1 Similarity Model Training

As aforementioned, given two interest keyword lists, we use the following features for training the similarity model:

- The cosine score of two keyword lists.
- The number of common keywords.
- The ratio of the number of common keywords to the number of keywords in the shorter list.
- The ratio of the number of common keywords to the number of keywords in the longer list.

With these features and target annotation by the Sina Weibo users of our game application, we are able to train the similarity model and evaluate its performance. In the experiments, our task is formalized as follows. Given a microblog user A and his/her two friends B and C , we get two pairs of users, e.g., (A, B) and (A, C) . Between the two pairs of the users, our model should determine which pair is more similar with each other than another pair. We collect 500 groups of annotations. In each group, a user selects two friends and annotates which one is more similar to him/her.

We can regard the problem as a classification problem. That is, given a user and his/her two friends, the pair that is more similar with each other is annotated as the positive instance ($y = 1$) while another is negative ($y = 0$). To prevent over-fitting in training, we apply 10-fold cross-validation. The most simple and efficient algorithm for classification is linear model, while the state-of-the-art classification algorithm is support vector machines (SVM). In this paper, we use

Table 1. Accuracy on training set

Method	Parameters	Accuracy
LIBSVM	b=1, the rest is default	93.2%
LIBLINEAR	s=6, the rest is default	92.0%
Ranking SVM	default	94.0%

LIBSVM [23] and LIBLINEAR [24] as the toolkit of SVM and linear model, both of which are the most widely used tools in natural language processing and machine learning. Table 1 shows the evaluation results.

The task can also be regarded as a ranking problem. That is, for a user as the input query, we rank his/her friends with more similar ones ranked higher. The problem can be addressed by learning-to-rank algorithms. Learning-to-rank algorithms can be divided into three approaches, including point-wise, pair-wise and list-wise. Our ranking task is naturally a pair-wise ranking problem. Therefore, we select the stat-of-the-art pair-wise algorithm, Ranking SVM [25], to solve our problem. Cross validation accuracy of Ranking SVM shows that the learned rule to the similarity model is effective (see Table 1).

The evaluation results show that the ranking assumption is more effective than the classification assumption for computing the similarity of user interests. This is not surprising because that our task is more like a ranking problem than a classification problem. For example, when a user tries to compare his/her friends in our game application, s/he is more concerned about the order. In the classification setting, however, it rigidly sets the preferred as 1 while the other as 0. This does not conform to reality comprehensively, because the distances between two friends will not always be 1.

4.2 Performance of the Framework on Sina Weibo

We apply our framework for measuring and visualizing the interest similarity on Sina Weibo. From 20th March, 2012, to 31th December, 2012, users on Sina Weibo have used our online system to visualize the interest similarities between themselves and their friends for more than 140,000 times. This phenomenon indicates our framework is effective and attractive.

It is usually difficult to quantify how well people welcome a new visualization technique; however, most users describe our visualization using “interesting”, “intuitionistic”, and “beautiful” in their microposts and on the message board of our system. A great deal of positive feedback indicates our visualization of interest similarity is satisfactory.

5 Conclusion

In this paper, we propose a novel framework for measuring and visualizing interest similarity between microblog users. By applying Ranking SVM method

on interest keywords extracted from microposts, we measure microblog users' interest similarity effectively, and the integrated word cloud visualization makes viewers comprehend the interest similarity clearly and intuitively. Besides, the interactive and attractive game we designed for collecting user annotations can help to train SVM-rank model constantly for better performance. Since applied on Sina Weibo, the largest microblogging service in China, our framework has attracted more than 140,000 times of usage in 9 months and has got plenty of positive feedback, which shows our framework is effective and encouraging.

We will consider the following work as the future research plan: (1) Our framework for measuring interest similarity does not hold the similarity transitivity. For example, when $\text{sim}(A, B) > \text{sim}(A, C)$ and $\text{sim}(A, C) > \text{sim}(B, C)$, our method does not guarantee that $\text{sim}(A, B) > \text{sim}(B, C)$, where $\text{sim}(X, Y)$ is the similarity score of user X and user Y derived from the similarity model. We will improve the framework for measuring interest similarity to hold the similarity transitivity. (2) It is obvious that the interests of most users will change over time. We will incorporate time factors into our framework. (3) We will learn to recommend relevant and useful information, such as users with similar interests and articles on relevant topics, according to the results of interest similarity measurement.

Acknowledgements. This work is supported by the Key Project in the National Science and Technology Pillar Program under Grant No. 2009BAH41B04 and the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative administered by the IDM Programme Office. The authors would like to thank Shiqi Shen for his help.

References

1. Banerjee, N., Chakraborty, D., Dasgupta, K., Mittal, S., Joshi, A., Nagar, S., Rai, A., Madan, S.: User interests in social media sites: an exploration with micro-blogs. In: CIKM 2009, pp. 1823–1826. ACM, New York (2009)
2. Viegas, F.B., Wattenberg, M., Feinberg, J.: Participatory Visualization with Wordle. IEEE Transactions on Visualization and Computer Graphics 15, 1137–1144 (2009)
3. Java, A., Song, X., Finin, T., Tseng, B.: Why we twitter: understanding microblogging usage and communities. In: WebKDD/SNA-KDD 2007, pp. 56–65. ACM, New York (2007)
4. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: WWW 2010, pp. 591–600. ACM, New York (2010)
5. Wu, S., Hofman, J.M., Mason, W.A., Watts, D.J.: Who says what to whom on twitter. In: WWW 2011, pp. 705–714. ACM, New York (2011)
6. Bakshy, E., Hofman, J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on twitter. In: WSDM 2011, pp. 65–74. ACM, New York (2011)
7. Zhao, D., Rosson, M.B.: How and why people Twitter: the role that micro-blogging plays in informal communication at work. In: GROUP 2009, pp. 243–252. ACM, New York (2009)

8. Krishnamurthy, B., Gill, P., Arlitt, M.: A few chirps about twitter. In: 1st Workshop on Online Social Networks, pp. 19–24. ACM, New York (2008)
9. Piao, S., Whittle, J.: A Feasibility Study on Extracting Twitter Users' Interests Using NLP Tools for Serendipitous Connections. In: PASSAT/SocialCom 2011, pp. 910–915. IEEE CS Press, New Jersey (2011)
10. Wu, W., Zhang, B., Ostendorf, M.: Automatic generation of personalized annotation tags for Twitter users. In: HLT 2010, pp. 689–692. ACL, Stroudsburg (2010)
11. Yamaguchi, Y., Amagasa, T., Kitagawa, H.: Tag-based User Topic Discovery Using Twitter Lists. In: ASONAM 2011, pp. 13–20. IEEE CS Press, New Jersey (2011)
12. Michelson, M., Macskassy, S.A.: Discovering users' topics of interest on twitter: a first look. In: AND 2010, pp. 73–80. ACM, New York (2010)
13. Paulovich, F.V., Toledo, F.M.B., Telles, G.P., Minghim, R., Nonato, L.G.: Semantic Wordification of Document Collections. Computer Graphics Forum 31, 1145–1153 (2012)
14. Cui, W., Wu, Y., Liu, S., Wei, F., Zhou, M.X., Qu, H.: Context-Preserving, Dynamic Word Cloud Visualization. IEEE Computer Graphics and Applications 30, 42–53 (2010)
15. Rivadeneira, A.W., Gruen, D.M., Muller, M.J., Millen, D.R.: Getting our head in the clouds: toward evaluation studies of tagclouds. In: CHI 2007, pp. 995–998. ACM, New York (2007)
16. Lohmann, S., Ziegler, J., Tetzlaff, L.: Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreicher, L., Palanque, P., Prates, R.O., Winckler, M. (eds.) INTERACT 2009 Part I. LNCS, vol. 5726, pp. 392–404. Springer, Heidelberg (2009)
17. Yu, L., Asur, S., Huberman, B.A.: What Trends in Chinese Social Media. arXiv:1107.3522v1 (2011)
18. A stacked model based on word lattice for Chinese word segmentation and part-of-speech tagging, <http://nlp.csai.tsinghua.edu.cn/thulac>
19. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: NAACL 2003, pp. 173–180. ACL, Stroudsburg (2003)
20. Liu, Z., Chen, X., Sun, M.: Mining the interests of Chinese microbloggers via keyword extraction. Frontiers of Computer Science in China 6, 76–87 (2012)
21. Joachims, T.: Optimizing search engines using clickthrough data. In: KDD 2002, pp. 133–142. ACM, New York (2002)
22. Halvey, M.J., Keane, M.T.: An assessment of tag presentation techniques. In: WWW 2007, pp. 1313–1314. ACM, New York (2007)
23. Chang, C., Lin, C.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 1–27 (2011)
24. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research 9, 1871–1874 (2008)
25. Joachims, T.: Training linear SVMs in linear time. In: KDD 2006, pp. 217–226. ACM, New York (2006)

Correlation Range Query

Wenjun Zhou and Hao Zhang

Statistics, Operations, and Management Science Department
University of Tennessee, Knoxville, TN 37996, USA
`wzhou4@utk.edu`
Department of Electrical Engineering and Computer Science
University of Tennessee, Knoxville, TN 37996, USA
`haozhang@utk.edu`

Abstract. Efficient correlation computation has been an active research area of data mining. Given a large dataset and a specified query item, we are interested in finding items in the dataset that are within certain range of correlation with the query item. Such a problem, known as the correlation range query (CRQ), has been a common task in many application domains. In this paper, we identify piecewise monotone properties of the upper and lower bounds of the ϕ coefficient, and propose an efficient correlation range query algorithm, called CORAQ. The CORAQ algorithm effectively prunes many items without computing their actual correlation coefficients with the query item. CORAQ also attains completeness and correctness of the query results. Experiments with large benchmark datasets show that this algorithm is much faster than its brute-force alternative and scales well with large datasets.

Keywords: Association Mining, Correlation Computing, ϕ Coefficient.

1 Introduction

The correlation range query (CRQ) problem has been a common task in many application domains. For example, in retail database marketing [1, 2], sellers recommend items to customers by analyzing their purchase histories. The general assumption is that highly correlated items are likely to be purchased together, and negatively correlated items tend not to be purchased simultaneously. As another example, when exploring proteins' functional linkages by analyzing genome sequencing data, it is hypothesized that proteins that function together are likely to evolve with some correlation with each other [3, 4]. With large and increasing quantities of data, it is essential to provide effective computational solutions to find correlated items efficiently and accurately.

Existing work [5–10] attempt to address the correlation computing problem in various ways, most of which target for finding the most correlated item pairs, either positively or negatively. A typical scenario is to find all item pairs whose correlation values are above a user-specified threshold, or a top-k list of the most highly correlated pairs. We name this “strong pair query.” However, in practice, sometimes we are not interested in the most highly correlated item

pairs. For example, if we know that purchases of pasta and tomato sauce are highly correlated even without bundle promotions, promoting such pairs of items would not be very effective. Instead, we are more interested in item pairs with moderate correlation values, so that bundle promotions are likely to add value to sales. Thus, we are motivated to develop a solution for correlation computing in a general setting—looking for correlation values within a given range.

As the reduced form of the Pearson’s correlation for binary data, the ϕ coefficient has been widely used to measure correlation in binary databases. Earlier work [7] has found that, although ϕ correlation does not have a monotone property, it has an upper bound, which is two-dimensional monotone. The computation cost of the upper bound is much lower than that of the exact value of ϕ . Along this thread of research, we identify an upper bound and a lower bound of the ϕ coefficient when a query item is specified, and show piecewise monotone properties of these bounds for one-way correlation search. Based on our mathematical proof, we propose a new algorithm, named CORAQ, to utilize the computational advantage of pruning by such bounds. Experiments with benchmark datasets show that this algorithm works effectively and efficiently.

The remainder of this paper is organized as follows. The CRQ problem is formally introduced in Section 2, and related work is also discussed. In Section 3, we explore the upper and lower bounds of the ϕ coefficient and illustrate the monotone property of these bounds. In Section 4, we describe our CORAQ algorithm to solve the CRQ problem in an efficient way, and discuss its correctness and completeness. Experimental results are presented in Section 5, using benchmark datasets that are widely known in the association mining research community. Finally, we draw conclusions in Section 6.

2 Range Query by Correlation

In this section, we describe the correlation range query problem and provide a brief overview of the related work.

2.1 Problem Definition

Given a large dataset \mathcal{D} of transactions involving a number of items, a specified query item Q and an interval of real numbers $[l, u]$, where the lower limit is l and the upper limit is u inclusive ($-1 \leq l \leq u \leq 1$). the CRQ problem aims to find all items in \mathcal{D} whose correlations with item Q fall in the range $[l, u]$. For example, if we want to find items whose correlation with Q is between 0.6 and 0.8, then we set $l = 0.6$ and $u = 0.8$. This formulation of the CRQ provides a more general solution for correlation range query than strong pair queries.

An example of the CRQ problem is illustrated in Figure 1. Consider a small dataset with ten transactions involving seven unique items (as shown in the table on the left). Given the query item is b , the brute-force method to solve this problem (as shown in the table on the right) is to calculate the correlation of each item with item b , and then check if the correlation falls into a designated

Range query example for Item b (brute-force)

TID	Item	Item Pair	Correlation	[0.6,0.9]	[-0.6,-0.1]
1	a,b,c,d	{b,a}	0.356	No	No
2	a,b,c	{b,c}	0.816	Yes	No
3	a,b,e	{b,d}	-0.583	No	Yes
4	b,c,f	{b,e}	0.089	No	No
5	a,b,c,g	{b,f}	-0.102	No	Yes
6	a,b,c	{b,g}	0.272	No	No
7	a,d,e				
8	a				
9	d,f				
10	d				

Fig. 1. An illustrative example of the CRQ problem

range. For example, given the range $[0.6, 0.9]$, the itemset we find is $\{c\}$. If the given range is $[-0.6, -0.1]$, the itemset we find is $\{d, f\}$.

2.2 Related Work

To effectively compute correlations for large datasets, Ilyas et al. propose the CORDS algorithm, which identifies highly correlated item pairs using a sampling-based approach [6]. Although this method is efficient, since it is based on sampling, it is always subject to false positives and false negatives. Xiong et al. has found a support-based upper bound of the ϕ coefficient for binary data to efficiently identify strongly correlated item pairs [5]. Accordingly, they further introduce an algorithm named TAPER to identify all of the item pairs whose correlations are above a given threshold [7]. Along this thread, the TOP-COP algorithm introduced in [11, 8] identifies the first k item pairs with the highest correlations. This algorithm does not require users to select a threshold, and therefore does not necessarily require good understanding of the data feature in advance. In spite of their effectiveness, however, the TAPER and the TOP-COP algorithms do not directly address the range query problem.

3 The ϕ Coefficient and Its Computational Properties

The Pearson's correlation coefficient is a widely used measurement of the correlation between two random variables. For binary variables, this coefficient is called the ϕ coefficient. As derived in [5], given two disjoint item sets A and B , the computation formula of their correlation, ϕ , can be written as:

$$\phi = \frac{\text{supp}(A, B) - \text{supp}(A)\text{supp}(B)}{\sqrt{\text{supp}(A)\text{supp}(B)(1 - \text{supp}(A))(1 - \text{supp}(B))}}, \quad (1)$$

where $supp(\cdot)$ represents the support of an item set [12]. More specifically, assume there are N transactions in the dataset and an item set A appears in N_A of them, then the support of item set A can be computed as $supp(A) = \frac{N_A}{N}$.

Before deriving the bounds of the ϕ coefficient and discussing their properties, we define $sro(A)$, the square root of odds for item A , to simplify notation. Note that $sro(A) \geq 0$ is true by definition.

Definition 1 (Square Root of Odds). *Given an item A with support $supp(A)$, its square root of the odds (SRO) is defined as:*

$$sro(A) = \sqrt{\frac{supp(A)}{1 - supp(A)}} \quad (2)$$

Lemma 1 (1-D Monotone Property of SRO). *$sro(A)$ is a monotonically increasing function of $supp(A)$.*

Proof. For any two items A_1 and A_2 such that $supp(A_1) < supp(A_2)$, we have $1 - supp(A_1) > 1 - supp(A_2)$. It follows that $\frac{supp(A_1)}{1 - supp(A_1)} < \frac{supp(A_2)}{1 - supp(A_2)}$, so

$$sro(A_1) = \sqrt{\frac{supp(A_1)}{1 - supp(A_1)}} < \sqrt{\frac{supp(A_2)}{1 - supp(A_2)}} = sro(A_2)$$

Without loss of generality, $sro(A)$ is monotonically increasing with the increase of $supp(A)$. \square

In the following, unless otherwise specified, Q denotes the query item, and T denotes another item, whose correlation with Q is to be assessed. We derive an upper bound and a lower bound of the ϕ coefficient, which help us reduce the search space.

3.1 The Upper Bound

As derived in [5], for any two items A and B with $supp(A) \geq supp(B)$, the upper bound of the ϕ coefficient for item pair $\{A, B\}$ is:

$$upper(\phi_{\{A,B\}}) = \sqrt{\frac{supp(B)}{supp(A)}} \sqrt{\frac{1 - supp(A)}{1 - supp(B)}} \quad (3)$$

Based on Equation (3), we derive the upper bound of the ϕ coefficient for the CRQ problem.

Lemma 2 (Upper Bound). *Given item pair $\{Q, T\}$ in the CRQ problem, the upper bound of the ϕ coefficient is:*

$$upper(\phi_{\{Q,T\}}) = \min \left\{ \frac{sro(T)}{sro(Q)}, \frac{sro(Q)}{sro(T)} \right\} \quad (4)$$

Proof. Since query item Q is given, its support $supp(Q)$ is a constant. According to Equation (3), for any item T , there are two cases:

- Case I: if $supp(T) \leq supp(Q)$, then

$$upper(\phi_{\{Q,T\}}) = \sqrt{\frac{supp(T)}{supp(Q)}} \sqrt{\frac{1 - supp(Q)}{1 - supp(T)}} = \frac{sro(T)}{sro(Q)} \leq \frac{sro(Q)}{sro(T)}. \quad (5)$$

- Case II: if $supp(T) > supp(Q)$, then

$$upper(\phi_{\{Q,T\}}) = \sqrt{\frac{supp(Q)}{supp(T)}} \sqrt{\frac{1 - supp(T)}{1 - supp(Q)}} = \frac{sro(Q)}{sro(T)} < \frac{sro(T)}{sro(Q)}. \quad (6)$$

Obviously, Equation (4) summarizes these two cases. \square

Lemma 3 (Piecewise Monotonicity of the Upper Bound). *For any item T in \mathcal{D} , the upper bound of the ϕ coefficient of item pair $\{Q, T\}$, $upper(\phi_{\{Q,T\}})$, is monotone with respect to $supp(T)$. Specifically,*

- If $supp(T) \leq supp(Q)$, $upper(\phi_{\{Q,T\}})$ is monotonically increasing with the increase of $supp(T)$.
- If $supp(T) > supp(Q)$, $upper(\phi_{\{Q,T\}})$ is monotonically decreasing with the increase of $supp(T)$.

Proof. If $supp(T) \leq supp(Q)$, then $upper(\phi_{\{Q,T\}}) = \frac{sro(T)}{sro(Q)}$. Given Q , $supp(Q)$ is a constant, and therefore $\frac{1}{sro(Q)}$ is a non-negative constant. By Lemma 1, $sro(T)$ is monotonically increasing with the increase of $supp(T)$. Thus, $upper(\phi_{\{Q,T\}})$ is monotonically increasing with the increase of the support of item T . Similar proof can be conducted when $supp(T) > supp(Q)$. \square

3.2 The Lower Bound

In this subsection, we derive a lower bound of the ϕ coefficient for the CRQ problem. Also, we identify a piecewise monotone property of the lower bound.

Lemma 4 (Lower Bound). *Given item pair $\{Q, T\}$ in the CRQ problem, the lower bound of the ϕ coefficient is:*

$$lower(\phi_{\{Q,T\}}) = \max \left\{ -sro(Q)sro(T), -\frac{1}{sro(Q)sro(T)} \right\}. \quad (7)$$

Proof. Because $supp(Q) + supp(T) - supp(Q, T) \leq 1$, we obtain $supp(Q, T) \geq supp(Q) + supp(T) - 1$. In addition, since $supp(Q, T) \geq 0$, we obtain $supp(Q, T) \geq \max\{0, supp(Q) + supp(T) - 1\}$. There are two cases:

- Case I: If $\text{supp}(T) \leq 1 - \text{supp}(Q)$, then the lower bound of $\text{supp}(Q, T)$ is 0. In this case, we obtain

$$\begin{aligned}
\phi_{\{Q,T\}} &= \frac{\text{supp}(Q, T) - \text{supp}(Q)\text{supp}(T)}{\sqrt{\text{supp}(Q)\text{supp}(T)(1 - \text{supp}(Q))(1 - \text{supp}(T))}} \\
&\geq \frac{0 - \text{supp}(Q)\text{supp}(T)}{\sqrt{\text{supp}(Q)\text{supp}(T)(1 - \text{supp}(Q))(1 - \text{supp}(T))}} \\
&= -\sqrt{\frac{\text{supp}(Q)}{1 - \text{supp}(Q)}} \sqrt{\frac{\text{supp}(T)}{1 - \text{supp}(T)}} \\
&= -\text{sro}(Q)\text{sro}(T) \geq -\frac{1}{\text{sro}(Q)\text{sro}(T)}. \tag{8}
\end{aligned}$$

The last inequality is true because when $\text{supp}(T) \leq 1 - \text{supp}(Q)$, we have $1 - \text{supp}(T) \geq \text{supp}(Q)$ and so $\frac{\text{supp}(T)}{1 - \text{supp}(T)} \leq \frac{1 - \text{supp}(Q)}{\text{supp}(Q)}$. Therefore,

$$\text{sro}^2(Q)\text{sro}^2(T) = \frac{\text{supp}(Q)}{1 - \text{supp}(Q)} \frac{\text{supp}(T)}{1 - \text{supp}(T)} \leq 1.$$

Since both $\text{sro}(Q)$ and $\text{sro}(T)$ are non-negative, it is straightforward to show that Inequality (8) holds.

- Case II: If $\text{supp}(T) > 1 - \text{supp}(Q)$, then the lower bound of $\text{supp}(Q, T)$ is $\text{supp}(Q) + \text{supp}(T) - 1$. in this case, we similarly obtain

$$\begin{aligned}
\phi_{\{Q,T\}} &> \frac{\text{supp}(Q) + \text{supp}(T) - 1 - \text{supp}(Q)\text{supp}(T)}{\sqrt{\text{supp}(Q)\text{supp}(T)(1 - \text{supp}(Q))(1 - \text{supp}(T))}} \\
&= \frac{-(1 - \text{supp}(Q))(1 - \text{supp}(T))}{\sqrt{\text{supp}(Q)\text{supp}(T)(1 - \text{supp}(Q))(1 - \text{supp}(T))}} \\
&= -\frac{1}{\text{sro}(Q)\text{sro}(T)} \geq -\text{sro}(Q)\text{sro}(T).
\end{aligned}$$

Obviously, Equation (7) summarizes these two cases. \square

Lemma 5 (Piecewise Monotonicity of the Lower Bound). *For any item T in \mathcal{D} , the lower bound of the ϕ coefficient for item pair $\{Q, T\}$, $\text{lower}(\phi_{\{Q,T\}})$, is monotone with respect to $\text{supp}(T)$. Specifically,*

- If $\text{supp}(T) \leq 1 - \text{supp}(Q)$, $\text{lower}(\phi_{\{Q,T\}})$ is monotonically decreasing with the increase of $\text{supp}(T)$.
- If $\text{supp}(T) > 1 - \text{supp}(Q)$, $\text{lower}(\phi_{\{Q,T\}})$ is monotonically increasing with the increase of $\text{supp}(T)$.

Proof. If $\text{supp}(T) \leq 1 - \text{supp}(Q)$, then $\text{lower}(\phi_{\{Q,T\}}) = -\text{sro}(Q)\text{sro}(T)$. Since Q is given, $\text{supp}(Q)$ is a constant, and thus $\text{sro}(Q)$ is a non-negative constant. By Lemma 1, $\text{sro}(T)$ is monotonically increasing with the increase of $\text{supp}(T)$. Thus, $\text{lower}(\phi_{\{Q,T\}})$ is monotonically decreasing with the increase of the support of item T . Similar proof can be conducted when $\text{supp}(T) > 1 - \text{supp}(Q)$. \square

3.3 Graphical Illustration of the Bounds

Figure 2 graphically illustrates Lemmas 3 and 5. Let us consider a CRQ with a user-specified query item Q , whose support value is $\text{supp}(Q) = q$. Assume there are items A, A', B and B' , whose support values are a, a', b and b' , respectively. These support values satisfy $a' \leq a \leq b \leq b'$, as shown in the figure.

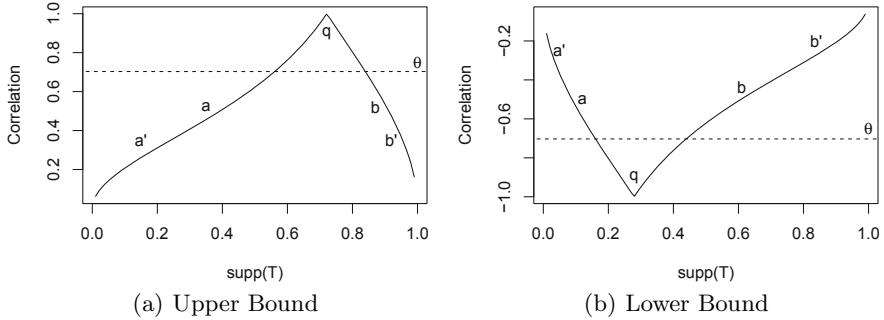


Fig. 2. An illustration of the bounds for a given query item

In Figure 2(a), if we find the upper bound $\text{upper}(\phi_{\{Q, A\}}) \leq \theta$, then according to Lemma 3, $\text{upper}(\phi_{\{Q, A'\}})$ is guaranteed to be no greater than $\text{upper}(\phi_{\{Q, A\}})$, thus, of course, no greater than the threshold θ . Without loss of generality, if we find any item A , whose support is less than q and $\text{upper}(\phi_{\{Q, A\}}) \leq \theta$, then for any item A' such that $\text{supp}(A') < \text{supp}(A)$, it is guaranteed that $\text{upper}(\phi_{\{Q, A'\}}) \leq \theta$. In other words, we can generate a one-dimensional pruning window to efficiently eliminate items that do not go beyond the correlation threshold θ . Similarly, if we identify an item B , satisfying $\text{supp}(B)$ is greater than q and the upper bound $\text{upper}(\phi_{\{Q, B\}})$ is no greater than the threshold θ , then any item B' satisfying $\text{supp}(B') \geq \text{supp}(B)$ can be safely pruned, since its upper bound $\text{upper}(\phi_{\{B', Q\}})$ is guaranteed to be no greater than $\text{upper}(\phi_{\{B, Q\}})$, according to Lemma 3. In Figure 2(b), similar discussions can be made.

3.4 The Search Space

For a CRQ, assume we are given a query item Q and a correlation range $[l, u]$, we obtain the following inferences regarding the reduced computational space.

Lemma 6. *If $l \geq 0$, we only need to calculate the correlations of item Q with item T that satisfies:*

$$\frac{l^2 \text{supp}(Q)}{1 - (1 - l^2)\text{supp}(Q)} \leq \text{supp}(T) \leq \frac{\text{supp}(Q)}{l^2 + (1 - l^2)\text{supp}(Q)}. \quad (9)$$

Proof. Since query item Q is given, its support $\text{supp}(Q)$ is a constant. For any other item T , there are two cases:

- Case I: if $supp(T) \leq supp(Q)$, then by Lemma 3, we only need to calculate the exact ϕ correlation coefficient for item Q with any item T satisfying

$$upper(\phi_{\{Q,T\}}) = \frac{sro(T)}{sro(Q)} \geq l$$

Solving $supp(T)$ from this inequality, we obtain

$$supp(T) \geq \frac{l^2 supp(Q)}{1 - (1 - l^2) supp(Q)}.$$

So we have

$$\frac{l^2 supp(Q)}{1 - (1 - l^2) supp(Q)} \leq supp(T) \leq supp(Q).$$

- Case II: if $supp(T) \geq supp(Q)$, then by Lemma 3, we only need to calculate the exact ϕ correlation coefficient for item Q with any item T satisfying

$$upper(\phi_{\{Q,T\}}) = \frac{sro(Q)}{sro(T)} \geq l$$

Solving $supp(T)$ from this inequality, we obtain

$$supp(T) \leq \frac{supp(Q)}{l^2 + (1 - l^2) supp(Q)}.$$

So we have

$$supp(Q) \leq supp(T) \leq \frac{supp(Q)}{l^2 + (1 - l^2) supp(Q)}.$$

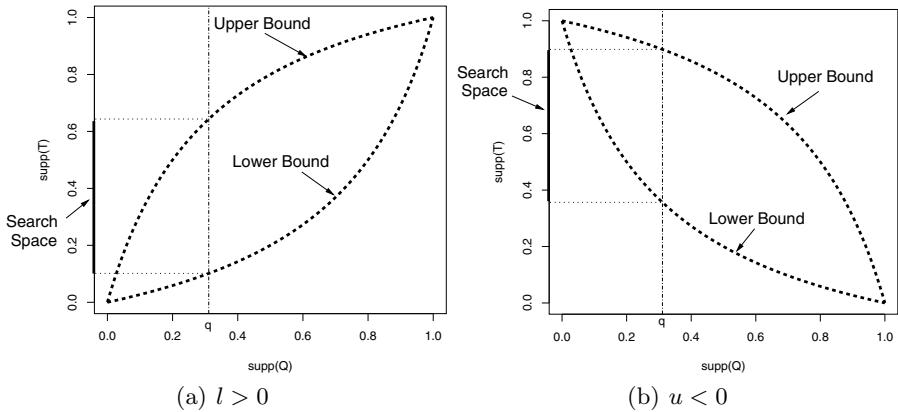
Obviously, Inequality (9) summarizes these two cases. \square

Lemma 7. *If $u \leq 0$, we only need to calculate the correlations of item Q with item T that satisfies:*

$$\frac{u^2(1 - supp(Q))}{supp(Q) + u^2(1 - supp(Q))} \leq supp(T) \leq \frac{1 - supp(Q)}{1 - (1 - u^2) supp(Q)}. \quad (10)$$

Proof. We can prove this Lemma in a similar fashion with Lemma 9. Therefore, the details are omitted. \square

Figure 3 illustrates the reduced search space for the CRQ problem. In each subfigure, the x-axis and y-axis are support values of query and target items, respectively. In particular, Figure 3(a) illustrates the reduced search space for the CRQ problem when $l > 0$. We can draw an upper bound curve and a lower bound curve, which correspond to Equations (4) and (7). In this illustration, we plot upper/lower bound curves using threshold $l = 0.5$. Given the query item Q , we can draw a vertical dash line corresponding to the query item's support. This dash line has intersections with the upper and lower bound curves, and the range between these two intersections is the search space for T . Similarly, Figure 3(b) illustrates the reduced computational space for the CRQ problem when $u < 0$.

**Fig. 3.** An illustration of the computation space

4 The CORAQ Algorithm

Based on our observation that the upper and lower bounds of the ϕ coefficient can be used to reduce the computation space, we propose a **CORrelation RAnge Query (CORAQ)** algorithm. The description of the CORAQ algorithm is shown in Algorithm 1.

Input	: Q : the query item; L : the list of all other items; l : the lower limit of correlation; u : the upper limit of correlation ($u \geq l$)
Output	: All items whose correlation with item Q are no greater than u and no less than l .
1	Find $\text{supp}(Q)$ and initialize $\text{minsupp} \leftarrow 0$, $\text{maxsupp} \leftarrow 1$;
2	if $l > 0$ then $\text{minsupp} = \frac{l^2 \text{supp}(Q)}{1 - (1-l^2)\text{supp}(Q)}$; $\text{maxsupp} = \frac{\text{supp}(Q)}{l^2 + (1-l^2)\text{supp}(Q)}$;
3	if $u < 0$ then $\text{minsupp} = \frac{u^2(1-\text{supp}(Q))}{\text{supp}(Q) + u^2(1-\text{supp}(Q))}$; $\text{maxsupp} = \frac{1-\text{supp}(Q)}{1 - (1-u^2)\text{supp}(Q)}$;
4	$S \leftarrow [\text{minsupp}, \text{maxsupp}]$;
5	foreach item T in L do
6	Find $\text{supp}(T)$;
7	if $\text{supp}(T) \in S$ then
8	Find $\text{supp}(Q, T)$ and calculate ϕ ;
9	if $l \leq \phi \leq u$ then add T to the output;
10	end
11	end

Algorithm 1. Correlation Range Query(Q, L, l, u)

The CORAQ algorithm differs from the brute-force method in that it adds a filtering process using the correlation bounds before calculating the exact correlation value. If the upper bound is already below the upper threshold, or if

the lower bound is already above the lower threshold, then there is no need to calculate the exact correlation value. This filtering process helps reducing computation cost because the calculation of the upper bound is much less costly than that of the exact correlation value. By adding the filtering process, for most of the item pairs we only need to compute the upper bound without calculating the exact ϕ value.

The completeness and correctness of the CORAQ algorithm can be explained by the following three facts. Firstly, all items other than the query item in the database are checked. Secondly, the filtering process only prunes item pairs based on the piecewise monotone property, when the upper bounds of ϕ are greater than u , or its lower bounds are less than l . Therefore, any item whose correlation coefficient with the query item is within the correlation range will be checked at some point. Finally, all items that are not checked before the search completes are not in the output list. All these are guaranteed by the theoretical analysis in Section 3.

5 Experimental Results

In this section, we present experimental results to evaluate CORAQ's performance. All experiments were performed on a PC with a 1.66 GHz CPU and 1 GB of RAM running the Microsoft Windows XP operating system.

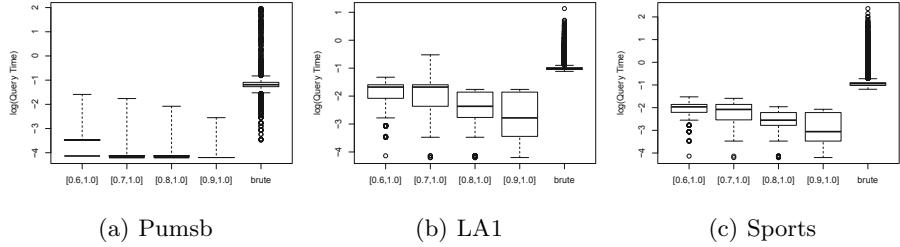
Table 1. Real-life dataset characteristics

Dataset	#Item	#Transaction	Source
Pumsb	2,113	49,046	FIMI
LA1	29,704	3,204	TREC
Sports	27,673	8,580	TREC

Table 1 summarizes benchmark datasets used in our experiments. **Pumsb** corresponds to binary versions of a census dataset, downloaded from (<http://fimi.ua.ac.be/data/>). It is often used as the benchmark to evaluate the performance of association rule algorithms over dense datasets. The **LA1** and **Sports** datasets are documents from the TREC collection (<http://trec.nist.gov>). Specifically, **LA1** contains news articles from the Los Angeles Times, whereas **Sports** contains articles from San Jose Mercury.

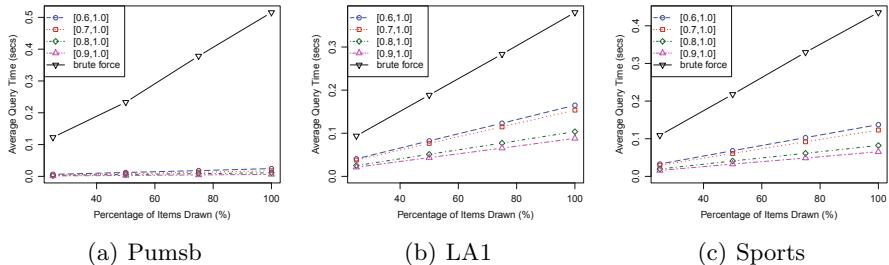
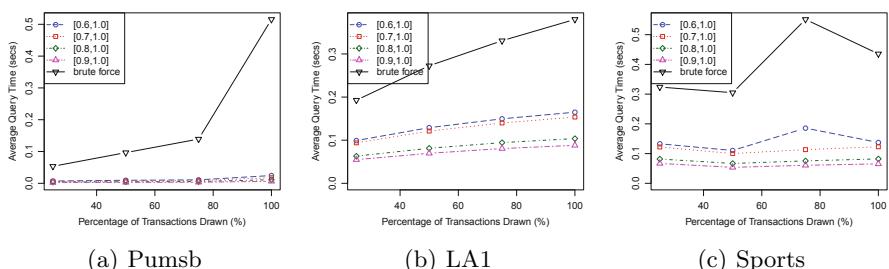
First, we compare the query time using our CORAQ algorithm (using different thresholds) with the brute-force approach. Figure 4 illustrates the distribution of the query time (in seconds, log scale) for individual items in each dataset. For each dataset, the execution time of our CORAQ algorithm is much faster than the brute-force approach, especially when the threshold is high. In addition, when the threshold increases, the execution time decreases accordingly.

Moreover, we also study the scalability of our CORAQ algorithm with respect to the number of items and the number of transactions, respectively. In order

**Fig. 4.** Distribution of query time (in seconds, log scale) for individual items

to see the difference in runtime over datasets with different sizes while keeping other data characteristics unchanged, we decrease each dataset's size using sampling techniques. Specifically, two different ways of sampling are employed: by sampling items, and by sampling transactions. We use smaller datasets, which are sampled at different ratios (75%, 50%, and 25%) and the original dataset in our scale-up experiments.

Figure 5 shows the average query time (in seconds) for each dataset when we sample data by items, whereas Figure 6 illustrates the average query time (in seconds) when we sample data by transactions. We observe that, in both cases, our CORAQ algorithm scales well with the dataset's size.

**Fig. 5.** Scalability study by sampling items**Fig. 6.** Scalability study by sampling transactions

6 Conclusions

In this paper, we propose a **CORrelation RAnge Query** (CORAQ) algorithm, which is based on the piecewise monotone property of the bounds of the ϕ correlation coefficient. The algorithm efficiently finds all items whose correlation with the query item is within a specified range, achieving completeness and correctness. As demonstrated by our experiments, our algorithm is much more efficient than the brute-force alternative for correlation range query task.

References

1. Seller, M., Gray, P.: A survey of database marketing. Technical report, I.T. in Business, Center for Research on Information Technology and Organizations, UC Irvine (1999)
2. Kamakura, W.A., Wedel, M., de Rosa, F., Mazzon, J.A.: Cross-selling through database marketing: a mixed data factor analyzer for data augmentation and prediction. *International Journal of Research in Marketing* 20, 45–65 (2003)
3. Pellegrini, M., Marcotte, E.M., Thompson, M.J., Eisenberg, D., Yeates, T.O.: Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proceedings of the National Academy of Sciences* 96(8), 4285–4288 (1999)
4. Xiong, H., He, X., Ding, C., Zhang, Y., Kumar, V., Holbrook, S.R.: Identification of functional modules in protein complexes via hyperclique pattern discovery. In: *Proceedings of the Pacific Symposium on Biocomputing*, pp. 221–232 (2005)
5. Xiong, H., Shekhar, S., Ning Tan, P., Kumar, V.: Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 334–343 (2004)
6. Ilyas, I.F., Markl, V., Haas, P.J., Brown, P., Aboulnaga, A.: CORDS: Automatic discovery of correlations and soft functional dependencies. In: *ACM SIGMOD International Conference on Management of Data*, pp. 647–658 (2004)
7. Xiong, H., Shekhar, S., Ning Tan, P., Kumar, V.: TAPER: A two-step approach for all-strong-pairs correlation query in large databases. *IEEE Transactions on Knowledge and Data Engineering* 18(4), 493–508 (2006)
8. Xiong, H., Zhou, W., Brodie, M., Ma, S.: Top-k ϕ correlation computation. *INFORMS Journal on Computing* 20(4), 539–552 (2008)
9. Zhou, W., Xiong, H.: Volatile correlation computation: A checkpoint view. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 848–856 (2008)
10. Zhou, W., Xiong, H.: Checkpoint evolution for volatile correlation computing. *Machine Learning* 83(1), 103–131 (2011)
11. Xiong, H., Brodie, M., Ma, S.: TOP-COP: Mining top-k strongly correlated pairs in large databases. In: *Proceedings of the 2006 IEEE International Conference on Data Mining*, pp. 1162–1166 (2006)
12. Tan, P.N., Steinbach, M., Kumar, V.: *Introduction to Data Mining*. Addison-Wesley (2006)

Real Time Event Detection in Twitter

Xun Wang¹, Feida Zhu², Jing Jiang², and Sujian Li^{1,*}

¹ Key Laboratory of Computational Linguistics (Peking University), MOE, China

² School of Information Systems, Singapore Management University

{xunwang, lisujian}@pku.edu.cn, {feidazhu, jingjiang}@smu.edu.sg

Abstract. Event detection has been an important task for a long time. When it comes to Twitter, new problems are presented. Twitter data is a huge temporal data flow with much noise and various kinds of topics. Traditional sophisticated methods with a high computational complexity aren't designed to handle such data flow efficiently. In this paper, we propose a mixture Gaussian model for bursty word extraction in Twitter and then employ a novel time-dependent HDP model for new topic detection. Our model can grasp new events, the location and the time an event becomes bursty promptly and accurately. Experiments show the effectiveness of our model in real time event detection in Twitter.

Keywords: HDP, Gaussian mixture, Twitter, event detection.

1 Introduction

Events usually refer to something abnormal, that is, something that rarely happens in normal situation. Event detection aims to find such abnormal phenomenon from collected data. Various methods have been proposed to detect events such as disease outbreaks, criticism and so on [1, 2].

When it comes to Twitter, events are topics that suddenly draw public attention, for example, music concerts, football matches and so on. Some topics are not events, although they are popular. For example, “iphone” and “ipad” have always been popular according to the trendy topics provided by twitter.com officially¹. They are talked about widely and repeatedly and show a high frequency everyday but are not bursty events. Neither are periodical topics events. For example, tweets on Friday usually talk about the coming of weekends while tweets on Monday usually complain about the beginning of a week of work. Such topics should not be regarded as events.

Several approaches have been proposed for event detection from tweets, such as wave analysis[3], topic model approach based on LDA[4], HDP[15] or text classification and clustering[5]. Existing approaches suffer from either failure in latent topics detection or inefficiency in involving topics models.

In this paper, we divide the event detection task into three steps. We firstly use a Gaussian mixture for bursty word candidate selection from huge tweets data. Our

* Corresponding author.

¹ <http://api.twitter.com/1/trends/>

candidate selection model considers both weekly effects and monthly effects. Further to decide whether a bursty word represents a new topic, we borrow the ideas of evolutionary clustering which focuses on detecting the dynamics of a given topic such as appearance, disappearance and evolution for new topics. We develop a novel time dependent HDP(td-HDP) model based on HDP model[6] for new event detection. Our model is based on the assumption that the data of Twitter forms a Markovian chain and each day's topic distribution is affected by the topic distribution of previous time. The number of events would naturally increase or decrease with the appearance and disappearance of new topics which can be detected in td-HDP. Finally, the location of event is detected from a CRF algorithm[16]. Usually there're three compulsory components of an event in Twitter: topic, location, the time it becomes bursty. Our model can grasp the three points promptly and accurately at the same time.

The rest of paper is organized as follows. Section 2 describes related works in event detection in Twitter. Bursty words detection is introduced in Section 3. The td-HDP model which aims at detecting new topics from bursty words is presented in Section 4. Location recognition is briefly described in Section 5. Experimental results are presented in Section 6 and we conclude the paper in Section 7.



Fig. 1. System architecture of our model

2 Related Works

2.1 Event Detection

Existing event detection methods usually treat words as signals. Some researches deal with words in the time domain. Kleinberg[7] used an infinite-state automaton to detect events, with events represented by the transitions of state. Fung et al.[8] developed the idea to estimate the appearance of words from binomial distribution and bursty words are detected with a threshold-based heuristic. There are some other methods that deal with words in the frequency domain[8] where traditional Discrete Fourier Transformation (DFT) is applied to convert the signals from the time domain into the frequency domain.

When it comes to Twitter, Weng et al.[3] applied the wavelet analysis to tweets. Words are converted into signals and corresponding signal auto-correlations are used to find non-trivial words. These non-trivial words are then clustered to form events. The problem of such kind of work is that words are treated as signals and it is hard to capture the latent topics within text. Topic models such as LDA have gained great success in these days for their clear and rigorous probabilistic interpretations for latent topic modeling. LDA has also been used for event detection task in Twitter[4]. The problem is, in LDA, the number of topics should be fixed in advance. So it's not suitable for real time event detection where the amount of data gradually grows, especially when the data is huge.

2.2 Evolutional Clustering and HDP

Evolutionary clustering is a relatively new research for topic detection, which aims to preserve the smoothness of clustering results over time, while fitting the data of each epoch. The work by Chakrabarti et al. [10] was probably considered as the first to address the problem of evolutionary clustering. They proposed a general framework of evolutionary clustering and extended two classical clustering algorithms to the evolutionary setting: (1) k-means clustering, and (2) agglomerative hierarchical clustering. The problem of evolutionary clustering is that the number of clusters stays the same over time. This assumption is obviously violated in many real applications.

Recently, HDP has been widely used in evolutionary clustering due to its capability of learning number of clusters automatically and sharing mixture components across different corpora. In HDP, each corpus is modeled by an infinite Dirichlet Process (DP) mixture model, and the infinite set of mixture clusters is shared among all corpora. Sethuraman [11] gave a stick-breaking constructive definition of DP for arbitrarily measurable base space and Blackwell and MacQueen[12] explained DP using the Polya urn scheme. The Polya urn scheme is closely related to the Chinese Restaurant Process (CRP) metaphor, which is applied on HDP demonstrating the ‘clustering property’ as the ‘distribution on partition’. Based on HDP, some algorithms of evolutionary clustering are proposed by incorporating time dependencies, such as DP-Chain, HDP-EVO and dynamic HDP et al [13], [14], [15].

3 Bursty Words Extraction

As stated, we firstly try to find bursty words which serve as candidates for new event detection in tweets. Bursty words are those whose frequencies severely increase in a short time. Words in tweets can be regarded as being drawn from a certain distribution. According to the central limit theorem, the frequencies of a word in each day can be approximately modeled by a Gaussian distribution. The parameters of distribution can be estimated from historical data. We use the records of data in tweets from Jan 2011 to Dec 2011 as the historical data. The frequency of each word is recorded in a 1×365 dimensional vector, denoting the number of appearance of certain word in each day of the year. Let $F_{x_i, t}$ denote the frequency of word x_i at day t . If x_i is not a bursty word, we assume that the distribution of $F_{x_i, t}$ should be almost the same as the mean frequency distribution in the past whole year. In addition, we also have to get rid of the periodical effect such as weekly effect and monthly effect. For example, topics in weekdays and weekends would be different and topics in different months should also be different. For example, when December comes, it is normal to find words such as ‘it is getting colder and colder’ in tweets. And these topics should not be regarded as new topics. We propose a mixture Gaussian distribution which can consider both weekly and monthly effect. For word x_i at time t , if it is not a bursty word, we assume that:

$$\begin{aligned}
F_{x_i t} \sim & a_{x_i t} \frac{1}{\sqrt{2\pi}\sigma_{x_i}^2} \exp\left[-\frac{(F_{x_i t} - \mu_{x_i})^2}{\sigma_{x_i}^2}\right] + b_{x_i w_t} \frac{1}{\sqrt{2\pi}\sigma_{x_i w_t}} \exp\left[-\frac{(F_{x_i t} - \mu_{x_i w_t})^2}{\sigma_{x_i w_t}^2}\right] \\
& + c_{x_i m_t} \frac{1}{\sqrt{2\pi}\sigma_{x_i m_t}} \exp\left[-\frac{(F_{x_i t} - \mu_{x_i m_t})^2}{\sigma_{x_i m_t}^2}\right] \quad a_{x_i t} > 0 \quad b_{x_i w_t} > 0 \quad c_{x_i m_t} > 0 \quad a_{x_i t} + b_{x_i w_t} + c_{x_i m_t} = 1
\end{aligned} \tag{1}$$

where w_t denotes whether time t is Monday, Tuesday,...Sunday, and m_t denotes whether time t is in Jan, Feb,...Dec. $w_t = [1, 2, \dots, 7]$ and $m_t = [1, 2, \dots, 12]$. The first term in Eq(1) concerns about the overall effect within a year for word x_i . The second term concerns about the weekly effect and the third term concerns about the monthly effect. All parameters in Eq(1) can be obtained from EM algorithm by maximizing likelihood estimate. But in this paper, we adopt the following approximation which largely cuts off running time.

$$\mu_{x_i} = \frac{1}{|T_1|} \sum_{t=1}^{|T_1|} F_{x_i t} \quad \sigma_{x_i}^2 = \frac{1}{|T_1|} \sum_{t=1}^{|T_1|} (F_{x_i t} - \frac{1}{|T_1|} \sum_{t=1}^{|T_1|} F_{x_i t})^2 \tag{2}$$

$$\mu_{x_i w_t} = \frac{1}{|T_{w_t}|} \sum_{t=1}^{|T_{w_t}|} F_{x_i t} \quad \sigma_{x_i w_t}^2 = \frac{1}{|T_{w_t}|} \sum_{t=1}^{|T_{w_t}|} (F_{x_i t} - \frac{1}{|T_{w_t}|} \sum_{t=1}^{|T_{w_t}|} F_{x_i t})^2 \tag{3}$$

$$\mu_{x_i m_t} = \frac{1}{|T_{m_t}|} \sum_{t=1}^{|T_{m_t}|} F_{x_i t} \quad \sigma_{x_i m_t}^2 = \frac{1}{|T_{m_t}|} \sum_{t=1}^{|T_{m_t}|} (F_{x_i t} - \frac{1}{|T_{m_t}|} \sum_{t=1}^{|T_{m_t}|} F_{x_i t})^2 \tag{4}$$

$|T_1|$ is 365. μ_{x_i} and $\sigma_{x_i}^2$ are the mean and variance of frequency of x_i in the whole year. Similarly, $\mu_{x_i w_t}$ and $\sigma_{x_i w_t}^2$ are the mean and variance of frequency of x_i at all w_t in a year. w_t could be Monday, Tuesday... This is the same case with $\mu_{x_i m_t}$ and $\sigma_{x_i m_t}^2$. Then $a_{x_i t}$, $b_{x_i w_t}$ and $c_{x_i m_t}$ can be learned through a maximum likelihood estimation. Experiments show that this approximation works well.

Next we get down to the bursty word selection. At time t , word with a daily frequency higher than the upper boundary of its 99% confidence interval according to Eq(1) would be selected as a bursty word. At each time t , we selected all words whose frequencies exceed the thresholds as bursty words which serve as candidates for new event detection in Section 4.

4 New Events Detection

In this section, we use the bursty words extracted from Section 3 to detect novel events in Twitter. We firstly describe Dirichlet Process and Hierarchical Dirichlet process, and then introduce our time dependent HDP model (tdHDP).

4.1 DP and HDP

A DP[11] can be considered as a distribution of probability measure G . Suppose a finite partition (T_1, \dots, T_K) in the measure space Θ and a probability distribution G_0 on Θ , we write $G \sim DP(\alpha, G_0)$ if $(G(T_1), \dots, G(T_K)) \sim Dir(\alpha G_0(T_1), \dots, \alpha G_0(T_K))$, where α is a positive concentration parameter and G_0 is called a base measure. Sethuraman [11] showed that a measure G drawn from a DP is discrete by the stick-breaking construction:

$$\pi_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l) = \beta_k \left(1 - \sum_{l=1}^{k-1} \pi_l\right), \quad \{\phi_k\}_{k=1}^{\infty} \sim G_0, \quad G = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k} \quad (5)$$

where δ_{ϕ_k} is a probability measure concentrated at ϕ_k . For convenience, we write $\pi \sim GEM(\alpha)$ if π is a random probability measure defined by Eq. (5).

A HDP[4] defines a distribution over a set of DPs. In HDP, a global measure G_0 is distributed as a DP with concentration parameter γ and base measure H . Then a set of measures $\{G_j\}_{j=1}^J$ is drawn independently from a DP with base measure G_0 . Such a process is described as:

$$G_0 \sim DP(\gamma, H), \quad G_j | \alpha_0, G_0 \sim DP(\alpha_0, G_0) \quad (6)$$

For each j , let $\{\theta_{ji}\}_{i=1}^{n_j}$ be independent and identically distributed (i.i.d.) random variables drawn from G_j . n_j observations $\{x_{ji}\}_{i=1}^{n_j}$ are drawn from the mixture model:

$$\theta_{ji} \sim G_j, \quad x_{ji} \sim F(x | \theta_{ji}) \quad (7)$$

where $F(x | \theta_{ji})$ denotes the distribution of generating x_{ji} . Equations (6) and (7) complete the definition of a HDP mixture model, whose graphical representation is shown in Figure 2(a).

According to Eq. (5), the stick-breaking construction of HDP can be represented as:

$$(\beta_k)_{k=1}^{\infty} \sim GEM(\gamma), \quad G_0 = \sum_{k=1}^{\infty} \beta_k \delta_{\phi_k}, \quad (\pi_{jk})_{k=1}^{\infty} \sim DP(\alpha_0, \beta), \quad G_j = \sum_{k=1}^{\infty} \pi_{jk} \delta_{\phi_k} \quad (8)$$

The corresponding graphical model is shown in Figure 2(b).

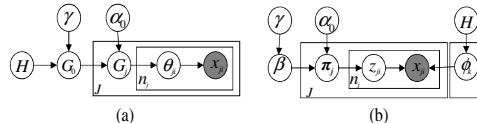


Fig. 2. HDP. (a) the original representation. (b)the stick-breaking construction.

4.2 td-HDP

We model the multiple correlated corpus with Dirichlet Processes with Markovian time dependency. Specially, at each time t , we use a DP to model the data from different time and then put the time dependency between different epochs based on the Markovian dependency. All DPs at different time G_t would share an identical base measure G , and G is drawn from $DP(\xi, H)$ where H is the base measure. According to Markovian assumption, G_t is not only affected by the overall base measure G , but also affected by G_{t-1} . The generation process of td-HDP is as follows:

1. Draw an overall base measure $G \sim DP(\varepsilon, G_0)$, which denotes the base distribution for all time data.
2. If t is the start point, draw the local measure $G_t \sim DP(\gamma^e, G)$ according to the overall measure G , else draw $G_t \sim DP(\gamma^e, (1-w^t)G + w^t G_{t-1})$, where $w^t = \exp(-c\Delta_{t,t-1})$. $\Delta_{t,t-1}$ denotes the exact time difference between epoch t and epoch $t-1$. In this paper $\Delta_{t,t-1}$ is set to one day. w^t is the factor that controls influence of topic distribution from previous time. c is the time controlling factor.
3. For each word $x_{t,i} \in D_t$ draw $\theta_{t,i} \sim g(\theta | G_t)$, $x_{t,i} \sim f(x | \theta_{t,i})$

According to the stick-breaking construction of DP, the overall base measure G can be represented with the following form:

$$G = \sum_{k=1}^{\infty} v_k \delta_{\phi_k}, v \sim GEM(\varepsilon) \quad G_t = \sum_{k=1}^{\infty} \pi_k^e \delta_{\phi_k}, \quad \pi^e \sim DP(\gamma^e, (1-w^t)v + w^t \pi^{e-1}) \quad (9)$$

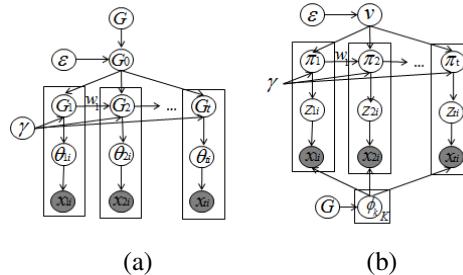


Fig. 3. (a)Graphical representation of our model (b) stick construction of our model

γ is sampled from a vague gamma prior which is set to be $Ga(10.0, 1.0)$.

4.3 Inference

We begin with the metaphor following Chinese Restaurant franchise for inference. At time t , the word collection D_t corresponds to a restaurant, and each word $x_{t,i} \in D_t$ corresponds to a customer. Each table t in the restaurant would have a dish k , which can be interpreted as the topic. All customers sitting around table m would enjoy dish k , meaning that these words are assigned to topic k . In the Chinese

Restaurant metaphor, customer x_{ti} sits at table p_{ti} while table p in restaurant D_t serves dish k_{tp} . MCMC sampling is used for td-HDP sampling.

We also need a notation for counts. $n_{tp\bullet}$ represents the number of customers in restaurant t at table p , and $n_{t\bullet k}$ represents the number of customers in restaurant t eating dish k . The notation m_{tk} denotes the number of tables in restaurant t serving dish k , $m_{t\bullet}$ denotes the number of tables in restaurant t , $m_{\bullet k}$ denotes the number of tables serving dish k in all restaurants, and $m_{\bullet\bullet}$ denotes the total number of tables in all restaurants. $n_{t\bullet k}$ denotes the number of customers in restaurant t having dish k and $m_{\bullet k}$ denotes the total number of tables serving dish k in all restaurants.

Sampling p. Due to the space limit, we would just show the sampling formula without derivation. The likelihood due to x_{ti} given $p_{ti} = p$ for some previously p is

$$f_{k_{tp}}^{-x_{ti}}(x_{ti}).$$

$$p(p_{ti} = p \mid p_{-ti}, \mathbf{k}) \propto \begin{cases} [(1 - w_{t-1})n_{tp\bullet} + w_{t-1}n_{t-1\bullet k_p} / m_{k_p}] \frac{E_{(x_{ti})}^{(k)} + \beta}{n_{\bullet k} + V\beta} & \text{if } p \text{ previously used} \\ \gamma \frac{1}{V} & \text{if } p = p^{\text{new}} \end{cases} \quad (10)$$

w_{t-1} is the influence factor from restaurant $t-1$, $E_{(x_{ti})}^{(k)}$ denotes the number of replicates of x_{ti} in topic k and β is the Dirichlet prior. If the sampled value of p_{ti} is p^{new} , then we can sample the topic of new table as follows $p_{kp^{\text{new}}}$

$$p(k_{p^{\text{new}}} = k \mid p, \mathbf{k}_{-q^{\text{new}}}) \propto \begin{cases} \frac{m_{\bullet k}}{m_{\bullet\bullet} + \varepsilon} \frac{E_{(x_{ti})}^{(k)} + \beta}{n_{\bullet k} + V\beta} & \text{if } k \in K \\ \frac{\varepsilon}{m_{\bullet\bullet} + \varepsilon} \frac{1}{V} & \text{if } k = k^{\text{new}} \end{cases} \quad (11)$$

Sampling k. Because the process of sampling t actually changes the component member of tables, we continue to sample k_{tp} for each table in the restaurant as follow:

$$p(k_{tp} = k \mid p, \mathbf{k}_{-tp}) \propto \begin{cases} \frac{m_{\bullet k}}{m_{\bullet\bullet} + \varepsilon} \frac{\Gamma(n_{\bullet k} + V\beta)}{\Gamma(n_{\bullet\bullet} + n_{tp\bullet} + V\beta)} \prod_{x_{ti} \in p_{tp}} \frac{\Gamma(E_{(x_{ti})}^{(k)} + E_{(x_{ti})}^{(p_{tp})} + \beta)}{\Gamma(E_{(x_{ti})}^{(k)} + \beta)} & \text{if } k \in K \\ \frac{\varepsilon}{m_{\bullet\bullet} + \varepsilon} \frac{\Gamma(V\beta)}{\Gamma(n_{tp\bullet} + V\beta)} \prod_{x_{ti} \in p_{tp}} \frac{\Gamma(E_{(x_{ti})}^{(p_{tp})} + \beta)}{\Gamma(\beta)} & \text{if } k = k^{\text{new}} \end{cases} \quad (12)$$

$E_{(x_{ti})}^{(p_{tp})}$ denotes the number of replicates x_{ti} at current table.

4.4 New Events Detection

In the two following situations, the event is regarded as new:

1. $k_{x_{ti}}$ =new. This means that the topic has never appeared before, so it is a new event .
2. $k_{x_{ti}}$ ≠ new and $k_{x_{ti}} \notin K_{t-i}$ $i \in [1,2,3]$. This means that even though $k_{x_{ti}}$ appeared before, but it did not appear in the past three days. So we can also regard $k_{x_{ti}}$ as new event.

Note each topic is represented by the top five words with largest probability.

5 Location Recognition

We also try to find the related locations for events. It is a traditional Named Entity Recognition task. In this paper, locations of an event is recognized through a CRF model[16]. The training data contains more than 1M tweets in Singapore, with several location names of events tagged. These events and their locations are manually selected and tagged using simple rules. For example, a car accident in Rochor Road is an event. Then “Rochor Road” in tweets of the same day such as “there’s a car accident at Rochor Road” or “I saw an accident in Rochor Road” would be tagged as a location of event. The feature template is unigram, current word w and four adjacent words, which are w, w+/-1 and w+/-2. Bigram is also used as an important feature.

6 Experiments

We use tweets of Singapore in one year as history data to decide the normal situation of words and tweets in May 2012 as test data. All these tweets are from more than 15k users who follow the 5 most popular political accounts in Singapore. We use the data from Jan. 2011 to Dec. 2011 as training data. Specifically, data from Jan. 2011 to Dec.2011 is used as history data for training of parameters in Gaussian mixtures described in Section 3 for bursty words extraction and data from Jan. 2011 to Mar.2011 for parameter tuning in 6.2. Data from Apr. 2012 to Jun. 2012 is used as test data.

6.1 Pre-processing

Words which contain too many replicates such as “hahahahaha”, “mmmmmmm” or “zzzzzz” or do not include valid characters such as “^_” are deleted. Moreover stop words are also deleted. Tweets that contain less than two words are also ignored.

6.2 Parameter Tuning

Firstly the time controlling factor in time dependant HDP needs to be tuned. We use data from Jan 2011 to Mar 2011 as training data and firstly construct the new event collection by manually selecting new events detected by different models. We asked five undergraduates in Singapore to find true events detected from event collection detected by Trendy Topic in twitter (list of hot topics given by twitter in a certain period of time), LDA model, tf-idf model, tdHDP($C=0$), tdHDP($c=1$) and tdHDP($c=5$). We collect 154 events in event collection. Then we experiment the td-HDP algorithm by setting c in the range from 0 to 5 with interval of 1. We compare the results with different c value with true event collection which was built previously according to manual evaluation and calculate accuracy for each experiment. We find that the accuracy drops sharply when c is set as a value larger than 2.0. Next, c is set in the range from 0 to 2.0 with interval of 0.2. Fig. 4 show the different value of accuracy with regard to different values of c . We find that the value of accuracy reaches their peak at around 1.2 and drops afterwards.

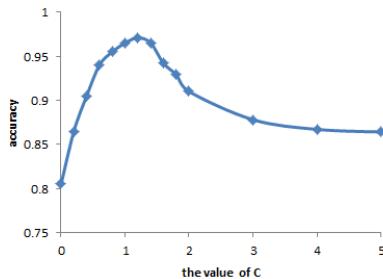


Fig. 4. Tuning the parameter C for tdHDP

6.3 Experimental Results

Due to the limitation of space, only events detected in May 2012 are shown in Table 1. We detect 33 events for May 2012 and 32 out of 33 are correct event detection according to manual evaluation. The wrongly detected events are marked in grey. As for Locations, “Online” means it’s an online topic with no related locations in real world. All 8 locations are correct event locations according to human judgment. We check the related tweets of the 9th one which is not a valid event. There are two different topics, “adobe cs6” and “cats in picasa”. They are mixed together because they share words such as “pics”.

Table 1. Events in May 2012

Date & Time	Events	Event Description	Locs
1st 4:55:19	labour happy holiday work	Labour day	Online
2nd 9:26:3	khj fans session meeting	Fan meeting of KHJ, a pop star	Online
2nd 15:36:42	Infinite word album brand	An album called Infinite is released	Online
4th 12:58:2	Kelantan match fans football	Football match, Kelantan vs LionsXII	Online
8th 13:33:9	whatsapp working wrong problem	Whatsapp, one app on smartphone	Online
8th 14:59:13	opera jap super junior dance	Super Junior's opera	Online
9th 7:34:19	ironman hulk captain avengers	Some popular movies	Online
9th 8:58:39	election hougang tony tan news	hougang election	Hougang
9th 5:29:34	cats cs6 faker picasa gallery	Not a valid event	N/A
12th 12:48:57	mothers day happy love mum	mothers' day comes	Online
13th 13:51:49	Alonso win Spanish Maldonado	Maldonado beat Alonso in Spanish Grand Prix	Online
13th 14:40:0	united man city beat play	Man City beat Man United	Online
13th 14:47:22	ferrari taxi accident driver dead	A ferrari taxi accident	Bugis
15th 6:57:2	diablo Funan Diablo iii fans	Funan Digital Mall released the game Diablo III	Online
16th 2:58:29	hougang pappng work election	hougang election	Hougang
16th 3:54:27	Zeng Guoyuan parrot police	Zeng Guoyuan's bird abused the police	Online
16th 8:38:31	speeding red driver beating lights	A speeding driver drove through the red light	Online
16th 12:25:54	pixie lott live topshop she	A famous singer Pixie Lott went to Top-Shop	Bugis
16th 12:39:10	England squad euro Neville	Gary Neville became England coach	Online
19th 20:26:18	goal Bayern Chelsea Thomas	Thomas Müller in the match: Bayern vs Chelsea	Online
19th 21:17:3	Chelsea win germans Bayern hope	Football match: Chelsea beats Bayern	Online
20th 12:28:35	Phua Chu kang Denise politics	hougang election	Hougang
20th 23:59:12	Gibb Robin dies singer cancer	Famous singer, Gibb Robin dies of cancer	Online
22nd 3:50:7	MBC concert google korean music	MBC's 'Korean Music Wave in Google'	Online
22nd 12:28:18	pxdkitty camera win blog world	To win a camera called Pxdkitty by blogging	Online
24th 15:7:5	thor thunder loki hammer rain	A movie called Thor	Online
26th 7:55:12	Joongki song today man running	Song Joongki, an actor in TV series	Marina
26th 14:38:1	worker party hougang partner win	hougang election	Hougang
27th 11:23:4	Taufik Rossa Batisah excited	Taufik & Rossa in Singapore TV show	Jalan
27th 13:58:54	Webber Mark Monaco wins	Mark Webber wins the game in Monaco	Online
28th 15:34:45	gaga lady concert stadium indoor	Lady GaGa's concert in stadium	Online
30th 9:48:33	sep 28th big bang coming	Pop band Big Bang in SG on Sept. 28th	Online

6.4 Evaluations

The evaluation of our model is conducted in three aspects. The first one is Timeliness, which represents whether our model can detect new events quickly. In addition, we evaluate the precision and recall of our model, which respectively denote the model's ability in detecting new events correctly and completely.

6.4.1 Evaluation of Timeliness

In tdHDP model, each word has a probability that is generated by a topic. Here we use the top word in the new topic to represent the event. The timeliness of an event is evaluated by the difference between the time when we detect the bursty word that represents the new topic and the peak appearance time of that word. Results of 32 events detected in May 2012 are shown in Fig. 4. Only 4 out of 32 events are detected after the frequency of bursty words arrives at its peak.

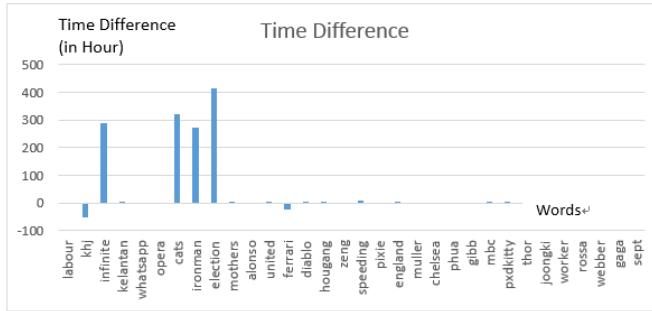


Fig. 5. The time difference between event detected time and peak time of bursty words

6.4.2 Evaluation of Precision and Recall

In this subsection, we evaluate the compare precision and recall of our model with existing models. The first baseline is standard HDP model which does not consider time dependent relations. LDA model and tf-idf model are also used as baselines.

Table 2. (a)Precision comparison with other models.(b)Recall comparison with other models.

Model	Precision
td-HDP	0.968
Standard HDP	0.890
LDA	0.841
tf-idf	0.806

(a)

Model	Recall
td-HDP	0.931
Standard HDP	0.832
LDA	0.797
tf-idf	0.710

(b)

As in 6.2, we build the event collection and manually judge the results of these methods. Then we compare the results of each model to the collection and get the Precision and Recall of each model. The results are shown in Table 2. Standard HDP achieves better results than LDA because HDP can model data more properly and topic number can increase and decrease naturally with the appearance and disappearance of topics. But in LDA, topic number has to be fixed in advance. Td-HDP

achieves best result and this further illustrates the necessity of adding Markovian time relation in topic modeling. Our approach enjoys the precision of 0.968 and a Recall of 0.931, demonstrating the effectiveness of our model.

7 Conclusion

We propose a novel event detection model for Twitter. A Gaussian Mixture model is constructed for word candidate selection in regard with periodic effect and a time-dependent HDP model is developed for new event detection from word candidates. Our model can deal with large amount of data effectively and detect events efficiently. Experiments show the good performance of our model.

Acknowledgements. This work is supported by NSFC programs (No: 61273278), NSSFC (No: 10CYY023), National Key Technology R&D Program (No: 2011BAH10B04-03), National High Technology R&D Program of China (No. 2012AA011101) and Singapore National Research Foundation under its IRC @ SG FI and administered by the IDM Programme Office.

References

1. Kulldorff, M., Mostashari, F., Duczmal, L., Yih, W.K., Kleinman, K., Platt, R.: Multivariate scan statistics for disease surveillance. *Statistics in Medicine* 26, 1824–1833 (2007)
2. Donoho, D., Jin, J.: Higher criticism for detecting sparse heterogeneous mixtures. *Annals of Statistics* 32(3), 962–994 (2004)
3. Weng, J., Lee, B.-S.: Event detection in twitter. In: ICWSM 2011, Barcelona, Spain (2011)
4. Diao, Q., Jiang, J., Zhu, F., Lim, E.-P.: Finding bursty topics from microblogs. In: ACL 2012, Jeju, Korea (2012)
5. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: WWW 2010, New York, USA (2010)
6. Teh, Y.W., Jordan, M.I., Beal, M.J., Blei, D.M.: Hierarchical Dirichlet Processes. *Journal of the American Statistical Association* 101(476), 1566–1581 (2006)
7. Kleinberg, J.: 2002 Bursty and hierarchical structure in streams. In: KDD 2002, New York, USA (2002)
8. Fung, G.P.C., Yu, J.X., Yu, P.S., Lu, H.: 2005 Parameter free bursty events detection in text streams. In: VLDB 2005, Trondheim, Norway (2005)
9. He, Q., Chang, K., Lim, E.-P.: Analyzing feature trajectories for event detection. In: SIGIR 2007, New York, USA (2007)
10. Chakrabarti, D., Kumar, R., Tomkins, A.: Evolutionary clustering. In: KDD 2006, Philadelphia, USA (2006), 2006
11. Sethuraman, J.: A constructive definition of Dirichlet priors. *Statistica Sinica* 2, 639–650 (1994)
12. Blackwell, D., MacQueen, J.B.: Ferguson distributions via Polya urn schemes. *The Annals of Statistics* 1, 353–355 (1973)
13. Xu, T., Zhang, Z.M., Yu, P.S., Long, B.: Dirichlet process based evolutionary clustering. In: ICDM 2008, Pisa, Italy (2008)
14. Ren, L., Dunson, D.B., Carin, L.: The dynamic hierarchical Dirichlet process. In: ICML 2008, Helsinki, Finland (2008)
15. Gao, Z.J., Song, Y., Liu, S.: Tracking and Connecting Topics via Incremental Hierarchical Dirichlet Processes. In: ICDE 2011, Hannover, Germany (2011)
16. John, L., Andrew, M., Fernando, P.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: ICML 2001, Williamstown, USA (2001)

A Data Cleaning Framework Based on User Feedback

Hui Xie, Hongzhi Wang, Jianzhong Li, and Hong Gao

Harbin Institute of Technology

xiehuixb@gmail.com, {wangzh, lijzh, honggao}@hit.edu.cn

Abstract. In this paper, we present our design of a data cleaning framework that combines interaction of data quality rules (CFDS, CINDS and MDs) with user feedback through an interactive process. First, to generate candidate repairs for each potentially dirty attribute, we propose an optimization model based on genetic algorithm. We then create a Bayesian machine learning model with several committees to predict the correctness of the repair and rank these repairs by uncertainty score to improve the learned model. User feedback is used to decide whether the model is accurate while inspecting the suggestions. Finally, our experiments on real-world datasets show significant improvement in data quality.

Keywords: data clean, user feedback, Bayesian decision, data quality rules.

1 Introduction

Real-life data is often dirty and costs billions of pounds to businesses worldwide each year. Dirty data have disastrous consequences for everyone [1]. A variety of approaches have been proposed for improving data quality including probabilistic, empirical, knowledge-based and logic-based approaches [6].

Most of the existing dependency-based data repair approaches [2,3] focus on providing fully automated solutions using different heuristics to select repairs that would introduce minimal changes to the data. But these heuristic algorithms can't ensure the correctness of the repairs and algorithm time complexity is too high, which could be risky especially for critical data.

In order to reduce the time complexity, we introduce a genetic model in section 2 to generate candidate repairs. To guarantee the correctness of the repairs, we create a machine learning model based on the Bayesian theory in section 3 and users (domain experts) are involved to confirm these repairs which can improve the learning model. Figure 1 shows the overall design of our framework.

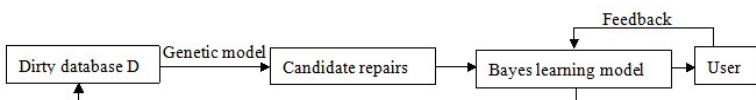


Fig. 1. Overall architecture

1.1 Related Work

Most closely related to our work is the line of research on inconsistent databases [2,3,4]. Consistent information is obtained from an inconsistent database focusing on the following approach: repair is to find another database that is consistent and minimally differs from the original database [7].

Data cleaning systems described in the research literature include the AJAX system [8] which provides users with a declarative language for specifying data cleaning programs, and the Potter's Wheel system [9] that extracts structure for attribute values and uses these to flag discrepancies in the data. Previous work on soliciting user feedback to improve data quality focuses on identifying correctly matched references in a large scale integrated data [10].

1.2 Contributions

We summarize our contributions as follows:

1. We present our genetic model, an optimization technique for data repair, which finds the best solutions to a problem by trying many different solutions and scoring them to determine their quality. (Section 2)
2. We incorporate user feedback in the cleaning process to enhance and accelerate existing automatic repair techniques while minimizing user involvement. (Section 3.1)
3. We designed a machine learning model that can meet our design goals of interactive response, fast convergence, and high accuracy. (Section 3.2)
4. The final contribution is an experimental study of our user feedback repair methods. Experiments on real-life datasets show the effectiveness of this framework to a better quality database with minimal user intervention. (Section 4)

2 Generating Repairs

In this section, we present our genetic model to generate candidate repairs. Instead of working with dataset, the genetic model attempts to select solutions that minimize the output of a cost function and satisfy the constraints at the same time.

Problem Definition: Given a database D comprising tables R_1, \dots, R_n , a set of constraints W defined on them and a cost function $Cost$. The task is to find the repair D for which $Cost(D)$ is minimum.

To solve this problem, we introduce three constraints [5]: (1) CFDs; (2) CINDs; (3) MDs and a cost function. For this problem, it is necessary to determine how a potential solution will be represented. We can find dirty tuples in D using constraints W and create a genetic model to find the best candidate repair which makes $Cost(D)$ minimum and satisfies constraints W .

Representing Solutions: We use $u(t, A_i)$ to denote the repair value of a given attribute $A_i \in attr(R_i)$ and use $u(t)$ to denote the repair value of a given tuple $t \in R_i$.

The following list:

$$[u(t_1), u(t_2), \dots, u(t_n)] \quad (1)$$

Represents a solution in which the tuple t_i take a repair value $u(t_i)$. For each $u(t_i)$, $u(t_i) = \sum_{i=1}^{i=n} u(t_i, A_i)$.

Cost Function: As mentioned above, $C = [u(t_1), u(t_2), \dots, u(t_n)]$ is used to represent a repair solution, thus the cost of the repair D is defined as

$$Cost(C) = \sum_{t \in C} u(t) \quad (2)$$

Where $u(t) = \sum_{A \in attr(R_i)} u(t, A)$.

For $u(t, A)$,

$$u(t, A) = sim(v, v') = 1 - \frac{dist_C(v, v')}{\max(|v|, |v'|)} \quad (3)$$

Where $dist_C(v, v')$ is an edit distance function. $u(t, A)$ is the score of an update u to modify $t[A] = v$ such that $t[A] = v'$.

Genetic Model: To generate candidate repairs, traditional heuristic algorithm is inefficient and the problem can be abstracted an optimization problem. As discussed above, a repair can be naturally coded to a sequence. Thus, we attempt use genetic algorithm to solve this problem. The genetic model is defined as follows.

$$GM = (C, E, P_0, M, \phi, \Gamma, \psi, T) \quad (4)$$

Where $C = [u(t_1), u(t_2), \dots, u(t_n)]$ is individual coding method, $E(C) = Cost(C)$ from Eq.2 is individual fitness evaluation function, $P_0 = [C_1, C_2, \dots, C_M]$ is initial population that can be obtained by assigning a random values to each entry, M is the size of the population, ϕ is selection operator, Γ is crossover operator, ψ is mutation operator and T is the terminating condition that is a threshold of the fitness function here.

Algorithm 1. *GenerateRepair($D, Cost, W$)*

Input: the database D , the cost function $Cost$ and the strains W

Output: the best solution $[u(t_1), u(t_2), \dots, u(t_n)]$

Begin

- 1: Identify dirty tuples in D using constraints W and stored in a *DirtyTuples* list
 - 2: For this *DirtyTuples*, create a set of random solutions $P_0 = [C_1, C_2, \dots, C_M]$ known as the initial population which are satisfied with this constraints
 - 3: **while** $\max\{E(C_i)\} < T$ **do**
 - 4: The cost function is calculated to get a ranked list of solutions
 - 5: Create the next population by roulette wheel selection ϕ , mutation ψ and crossover Γ
 - 6: **Done**
 - 7: Choose the best solution $[u(t_1), u(t_2), \dots, u(t_n)]$
-
- End**
-

3 Feedback-Based Data Repairing

The model in Section 2 generates candidate repairs, but it cannot ensure their correctness. To improve the quality of repairs, we introduce the novel users' feedback mechanism to confirm these repairs.

In this section, first we propose a Bayesian machine learning model (Section 3.1), which is suitable for embed feedback. For uncertain repairs, users give feedback and these responses constitute a new training set to improve the learned model. Then we use this model to confirm candidate repairs and use these repairs to repair dirty database (Section 3.2).

3.1 Learning Model

The goal of creating this model is to verify the correctness of these repairs generated in Section 2. This machine learning model is based on the Bayesian theory. In this model, for each attribute $C_i \in \text{attr}(R)$, we create a Bayes classifier M_{C_i} with several classifiers as committees for voting using training data T . To improve this model, we select n candidate repairs based on uncertainty score for labeling and user's responses for these repairs to construct a new training set to retrain the model.

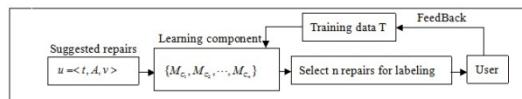


Fig. 2. User feedback model based on Bayesian theory

In the input, each suggested repair is in form of $u = < t, C, v >$, where t is tuple, C is an attribute and v is the suggested repair value for $t[C]$. The meanings of a user's response $r \in \{1, 0, -1\}$ is as follows.

- (1) $r = 1$ means that the value of $t[C]$ should be v .
- (2) $r = -1$ means that v is not a valid value for $t[C]$ and we need to find another repair.
- (3) $r = 0$, which means that $t[C]$ is a correct value and there is no need to generate more repairs for it.

For a repair $u = < t, C, v >$, we create a Bayes classifier M_C to give its response r using $p(r/u)$. The flow is illustrated in the Fig.2. First, given a set of training data T in the form $< u, r >$, we learn a Bayes classifier M_{C_i} with N committees for each attribute $C_i \in \text{attr}(R)$ through partitioning T into N disjoint sets. Second, we select n repairs from suggested repairs for labeling based on uncertainty score which is generated by measuring the disagreement amongst the predictions it gets from a committee of classifiers. Finally, users give their feedback that obtains a new training set to improve the predictive accuracy of learning component.

3.2 Cleaning Algorithm

In this section, we combine the genetic model with the machine learning model and introduce the complete data cleaning algorithm. The goal of this algorithm is to generate more accurate repairs by efficiently involving users to guide the cleaning process.

Algorithm 2. *UFeedBackRepair(D, Σ, T)*

Input: Dirty database D , cleaning rules Σ , Training data T

Output: Database repair D'

Begin

- 1: Train a set of Bayes classification models $\{M_{C_1}, M_{C_2}, \dots, M_{C_n}\}$ using T
- 2: Identify dirty tuples in D using Σ and stored in a *DirtyTuples* list.
- 3: **while** dirty tuples exist **do**
- 4: Calling *GenerateRepair()* to generate update in the form $< t, A, v >$ for all $t \in \text{DirtyTuples}$ and $A \in \text{attr}(R)$
- 5: The Bayes machine learning model predicts for suggested repairs
- 6: **While** user is not satisfied with the learner predictions **do**
- 7: The learning model selects n repairs for labeling
- 8: The user interactively gives feedback on the suggested repairs
- 9: The newly labeled repairs are added to the training dataset T and the learner is retrained
- 10: Generate new predictions for suggested repairs
- 11: **done**
- 12: The machine learning model decisions are applied to the database.
- 13: Check for new dirty tuples and generate repairs

13: **done**

End

4 Experimental Results

In this section, we present experimental results of our user feedback repair techniques. We use the percentage of the total number of correct repair as accuracy to evaluate the effectiveness of proposed methods.

Experimental Setting: We used two real-life data sets referred to as HOSP¹ and DBLP². All algorithms were implemented in Python and all experiments were run on a machine with an Intel(R) Core(TM) i3-2120 (3.30GHz) CPU and 4GB of memory.

¹ <http://www.hospitalcompare.hhs.gov/>

² <http://www.informatik.uni-trier.de/~ley/db/>

Experimental Results

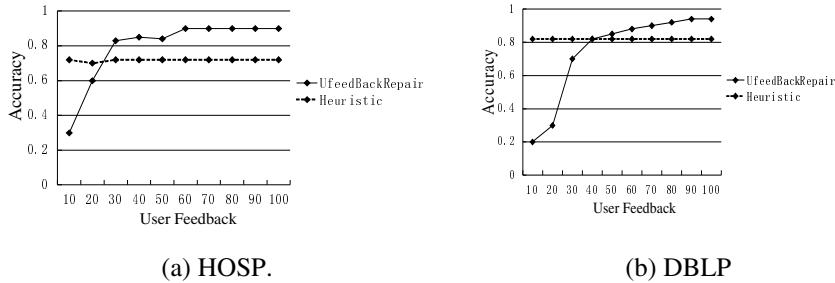


Fig. 3. Effectiveness of user feedback

In Figure 3, we show the progress in improving the quality against the number of verified repairs. The heuristic algorithm described in [2] for automatic data repair. The feedback is reported as a percentage of the total number of suggested updates through the interaction process to reach the desired clean database. The results show that our framework achieves superior performance in different datasets; for HOSP dataset, the framework gain about 83% improvement with 30% efforts. For DBLP dataset, about 82% quality improvement was gained with 50% efforts. The heuristic method repairs the database without user feedback; therefore, it produces a constant result. The accuracy of the heuristic approach is attained by *UFeedBackRepair* with about 30% user effort.

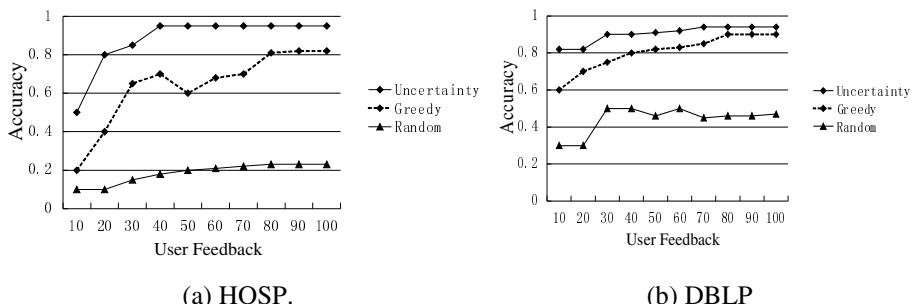


Fig. 4. Performance of user feedback

We evaluate the overall performance of the learning model based on uncertainty score by comparing its speed of convergence to the peak accuracy with other selection method.

- (1) *Random*: For suggested repairs, we randomly select n instances for labeling.
- (2) *Greedy*: We select n repairs according to their score from Eq. 3.

The experimental results are shown in Fig. 4. For both datasets, *Uncertainty* outperforms random and greedy selection significantly in accuracy. This is because the *Uncertainty* method finds the most beneficial repairs that are more likely to improve the learning model. *Uncertainty* is able to achieve a peak accuracy of 95% for the HOSP and 94% for the DBLP dataset. The same number of instances selected randomly, achieve accuracy of just 30% and 50% respectively for the HOSP and DBLP datasets. The greedy selection gets accuracy of 80% and 90% respectively for the HOSP and DBLP datasets.

5 Conclusion and Future Work

In this paper, we have proposed a framework which combines interaction of CFDS, CINDS and MDs with user feedback through an interactive process. The main novelty of this framework is to design an active learning algorithm that can meet our design goals of interactive response, fast convergence, and high accuracy.

For future work, we intend to extend our results to cover semi-structured data as well; specially, we intend to develop schemes for repairing constraint violations in the context of XML data integration.

Acknowledge. This paper was partially supported by NGFR 973 grant 2012CB316200, NSFC grant 61003046 and NGFR 863 grant 2012AA011004. Doctoral Fund of Ministry of Education of China (No.20102302120054). the Fundamental Research Funds for the Central Universities(No. HIT. NSRIF. 2013064)

References

1. Herzog, T.N., Scheuren, F.J., Winkler, W.E.: Data Quality and Record Linkage Techniques. Springer, Heidelberg (2007)
2. Bohannon, P., Fan, W., Flaster, M., Rastogi, R.: A cost-based model and effective heuristic for repairing constraints by value modication. In: ACM SIGMOD, pp. 143–154 (2005)
3. Cong, G., Fan, W., Geerts, F., Jia, X., Ma, S.: Improving data quality:consistency and accuracy. In: VLDB, pp. 315–326 (2007)
4. Lopatenko, A., Bravo, L.: Efficient approximation algorithms for repairing inconsistent databases. In: ICDE, pp. 216–225 (2007)
5. Fan, W., Geerts, F.: Foundations of Data Quality Management. In: Synthesis Lectures on Data Management (2012)
6. Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques. Springer (2006)
7. Greco, G., Greco, S., Zumpano, E.: A logical framework for querying and repairing inconsistent databases. IEEE Trans. Knowl. Data Eng. 15(6), 1389–1408 (2003)
8. Galhardas, H., Florescu, D., Shasha, D., Simon, E., Saita, C.: Declarative Data Cleaning: Language, Model and Algorithms. In: VLDB (2001)
9. Raman, V., Hellerstein, J.M.: Potter’s Wheel: An Interactive Data Cleaning System. In: VLDB (2001)
10. Jeery, S.R., Franklin, M.J., Halevy, A.Y.: Pay-as-you-go user feedback for dataspace systems. In: ACM SIGMOD, pp. 847–860 (2008)

Finding Critical Blocks of Information Diffusion in Social Networks

Ende Zhang^{1,2}, Guoren Wang², Kening Gao¹, and Ge Yu^{1,2}

¹ Computing Center, Northeastern University, China

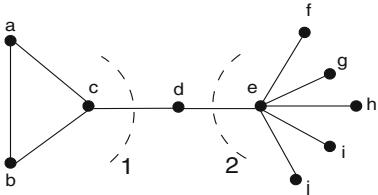
² College of Information Science & Engineering, Northeastern University, China
`{zed,gkn}@cc.neu.edu.cn, {wanggr,yuge}@mail.neu.edu.cn`

Abstract. The diffusion of information is taking place every place and every time over the Internet. The widely used web applications of online social networks, have many benefits to serve as a medium for fast, widespread information diffusion platforms. While there is a substantial works on how to maximize the diffusion of useful information, there are many misinformation diffusing on social networks. How to control the misinformation diffusing efficiently with the smallest cost is still a big challenge. We tackle this challenge by reducing the problem to finding the critical blocks. The critical blocks are the sets of nodes that partition the whole network evenly at a small cost, and we believe they play a key role during the process of diffusion. We prove such problem of finding critical blocks is NP-complete and therefore an exact solution is infeasible to get. A simple but effective solution is proposed by the following steps: first we convert a social network graph into a Laplacian matrix, then we compute its Fiedler Vector, which has been proved to have good properties, with the help of Fiedler Vector, we develop some heuristic algorithms to find critical blocks. We also perform lots of experiments both on synthetic data and real world datasets of Twitter, the experimental results show that our algorithm is effective and efficient both on synthetic data and real world data.

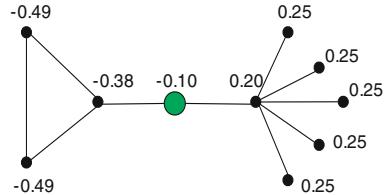
1 Introduction

The Web media, which generates and propagates huge amounts of information, is generally admitted as the fourth media after newspapers, broadcasting and TVs. The widely used web applications, especially the online social networks, such as micro-blog (e.g. Twitter) and SNS (e.g. Facebook), have many benefits to serve as a medium for fast, widespread information diffusion platforms. These platforms provide faster access than traditional mass media, including almost every aspects of events, vary from national policy to stars' privacy. But among all the information, there are many misinformation [6]. So it is desired if such diffusion of misinformation can be prevented at a small cost.

But unfortunately, the process of the diffusion is very complicated. And it is impossible to tell whom to be controlled directly. We focus on the study of the information diffusion over online social networks, we call them diffusion



(a) A simple example of network structure



(b) The Fiedler Vector's values corresponding with the network in Fig 1(a)

Fig. 1. An example of network structure and its corresponding Fiedler Vector's values

networks. In this work, we treat the problem of controlling the diffusion as to find the critical blocks of diffusion network. The critical blocks are sets of nodes in the network, and they play an important role in network structure, as illustrated in Fig 1(a). At first glance, node e has the largest degree, so it is a critical node in the network, if we remove node e (and its attached edges), it will dramatically change the network structure. We call such nodes with large degree is trivial critical nodes, for it is easy to find and we consider such nodes compose *trivial critical blocks*. It is should be noticed that node d is another critical node, since it is a bridge connecting the left part (left to dotted line 1) and the right part (right to dotted line 2), if we remove the node d , the network will be divided into two parts, and the network structure will also have a great change. Although node d only has a small degree, it is a critical node of the network. Removing node d can make the diffusion be confined in left part (or right part), for example, node a, b, c may form a community, node e, f, g, h, i, j may form another community, and node d may join both communities, if there is no node d , the information in left community will not flow to the right community, and vice versa. We call node d composes a *non-trivial critical block*, in this paper, we mainly focus on non-trivial critical blocks.

It should be mentioned the critical blocks in network is similar to *structural holes*[7] a little, which refer to the absence of ties between two parts of a network. The concept of structural hole is developed by sociologist Ronald Burt. And some following works [2, 5, 14] focus on how to use the structural holes to enhance ones competitiveness through person to person interaction. To the best of our knowledge, none of the existing work formally addressed the problem of finding the critical blocks to limit the diffusion at a small cost.

In this paper, we present an algorithm called FCB (Finding Critical Blocks) to find critical blocks of the network effectively. We consider if we can affect these blocks, we can interrupt the diffusion with smaller cost. But it can be seen, as the number of nodes in network increases, the structure of the network will become very complex too, so an effective and efficient algorithm is needed. For simplicity, we assume that the network structure stays stable and does not change over time. We implement our algorithm based on the spectral graph theory [21] with following steps: first we build a Laplacian matrix based on the structure of

the network; then we compute its eigenvectors corresponding with some smallest eigenvalues, and we show the relationship between the critical blocks and these eigenvectors, thus we can efficiently locate the critical blocks under the help of these eigenvectors. Since computing the eigenvalues and eigenvectors is a mature technology, we can easily solve them and we develop some heuristic algorithms based on them.

The rest of this paper is organized as follows. Section 2 introduces some related works. Section 3 gives the preliminaries. In Section 4 we formulate this problem and prove its NP-completeness and give our solution of finding critical blocks in details. In Section 5, our experimental results are reported. We make a conclusion and point out some future research directions in Section 6.

2 Related Work

The online social network analysis has attracted many researchers [1, 6, 9, 11, 15]. Among all the research directions, the research about information diffusing and tracking has received significant attention [6, 9, 11, 15]. In their studies, P. Pinto et al. [8] tried to locate the source of diffusion in network, J. Leskovec et al. [15] tried to track the memes of the web media. M. G. Rodriguez et al. [11] tried to infer the networks of diffusion and influence, they proved it is a NP-hard problem and gave a greedy algorithm called NetInf to solve it.

There is also a research area related to our work called social influence analysis, it has various real word applications, for example, influence maximization in viral marketing [12]. How to identify the influential users has received significant attention. Early works focused on heuristics using node degree and distance centrality [16]. The works in [15] improved the heuristics to detect influential nodes and gave solutions to such problem, they showed that this problem has a submodularity property and give their solution. Budak et al. [6] tried to limit the spread of misinformation using some heuristic greedy algorithms.

Our method is based on the Laplacian matrix theory of network graph, which is also called spectral graph theory. There are extensively research about this theory [21]. Laplacian matrix and its variations are useful tools in clustering [18] and graph partition [17, 19]. [21] gave detailed descriptions about the theory and applications. [9] is Fielder's work, which showed that the second smallest eigenvalue of Laplacian matrix can represent the algebraic connectivity and gave some properties. [18] gave detailed implementations of the spectral clustering, both showed good performance on clustering. The works in [17] and [19] both used Laplacian matrix theory to partitioned a network.

3 Preliminaries

A social network can usually be modeled as an undirected graph $G = (N, E)$, where N represents the nodes, and E represents the edges connecting N . $n_u \in N$ and $n_v \in N$ are said to be neighbors if there is an edge $e_{u,v}$ connecting them, noted as $E(u,v) = 1$. In the context of social network, N can be treated as

users and E can be treated as relationship between them, such as friendship, or colleague relationship.

Given a subset $S \subseteq N$ of a graph, we denote its complement $N \setminus S$ as \bar{S} . Generally, if there exist k nonempty connected sets S_1, S_2, \dots, S_k , they compose a partition of the graph G if $S_i \cap S_j = \emptyset$ and $\bigcup_{i=1}^k S_i = N$.

The Laplacian matrix L is defined as:

$$L = D - A$$

where A is the adjacency matrix of the graph and D is its diagonal matrix. L has many properties, the reader can refer [3], here we list some helpful properties to better understand our algorithm.

Proposition 1. *L is symmetric and positive semi-definite. And for any vector x ,*

$$x^T L x = \frac{1}{2} \sum_{i,j} (x_i - x_j)^2$$

Proposition 2. *There exists at least one 0 eigenvalue of L and its corresponding eigenvector is the constant vector $\mathbf{1} = (1, 1, \dots, 1)^T$.*

Proposition 3. *L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n, \lambda_2 > 0$ if and only if the graph G is connected.*

Proposition 4. *The multiplicity of 0 as an eigenvalue of L is equal to the number of disconnected components of G .*

4 Our Approach

In this section, firstly we will describe our problem formally, then we propose our algorithm to solve the problem of finding critical blocks effectively and efficiently.

4.1 Problem Formulation

While there has been a substantial amount of work in the field of influence maximization, how to restrict the misinformation diffusion has not received much attention. We regard this problem as to find the critical blocks in the diffusion network, critical blocks can be seen as some subgraphs which contain “influential” nodes. Here we give a descriptive definition of a critical block.

Critical block. Critical block is a set of nodes, and these nodes can partition the whole graph evenly at a small cost.

Considering this problem carefully, we can see such critical blocks should fit three conditions: the first condition is critical blocks should partition the whole network into several disconnected components, such that the information cannot diffuse from one part to another; the second one is that any critical block itself should be small enough, that means we can influence diffusion without affect

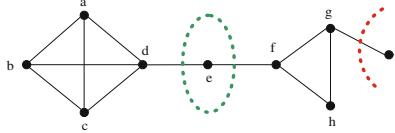


Fig. 2. A bad algorithm will give a bad critical block

too much nodes; the third one is the partition caused by critical blocks should be “balanced”, which means none of these disconnected components should be too small nor too big, see Fig 2. We can see if we choose node i to compose a critical block and cut the rightmost edge, we can partition the whole network into two parts and the cost is small, but the influence to the whole network is small too. So the rightmost node i is a “bad” critical block, instead, node e composes a “good” critical block for it satisfies the above three conditions. It should be noticed that it does not mean “balanced” parts have equal size.

How to evaluate a critical block ‘good’ or ‘bad’? There may not exist a ground truth to measure the results, but at first sight, a critical block partitions the whole graph into two subsets S and \bar{S} under the constraints that the critical block size (denoted as $CBS(S, \bar{S})$) itself should be small, then the problem to find the critical block is to find a partition which minimizes the $CBS(S, \bar{S})$. But as illustrated in Fig 2, if we only want to find a partition which minimizes $CBS(S, \bar{S})$, it may soon sink into the problem that a critical block separates one individual node from the rest of the graph. What we want is to partition the graph evenly, so either part should be “large enough” to balance the separation. Inspired by the RationCut [17] and Ncut [19], we give a criterion to measure the results as:

$$\operatorname{argmin}_{i \in S, j \in \bar{S}} \left(\frac{E(i, j)}{|S|} + \frac{E(i, j)}{|\bar{S}|} \right) \quad (1)$$

Unfortunately, finding such critical blocks is NP-Complete.

Theorem 1. *Finding a critical block defined in equation (1) is NP-complete.*

Proof. To prove the NP-completeness, we have to prove two things, first we have to prove this problem belongs to NP class, then we have to prove it via reduction from a known NP-complete problem. It is obvious that this problem \in NP, since a nondeterministic algorithm needs only to test a found critical block satisfying the constraints in polynomial time. Now we show that this theorem can be proved via reduction from a known NP-complete problem of Partition problem. Given an instance of Partition problem: find a partition into two subsets S_1, S_2 such that $\max(\sum(S_1), \sum(S_2))$ is minimized. Given a graph, if we can partition it into two parts P_1, P_2 evenly, we can assign a node i in P_1 corresponding to an element in S_1 , and assign a node j in P_2 corresponding to an element in S_2 , so if the finding critical block can be solved efficiently, then the Partition problem

must be solvable. Also we can see the assignment can be done in polynomial time, so finding critical block is a NP-complete problem.

Since finding critical blocks is a NP-complete problem, it is infeasible to get an exact solution, and we develop an effective way to find them approximately.

In this paper, we find all critical blocks hierarchically, we first find a critical block which partition the whole network into two parts, then we find other critical blocks of the two partitioned parts, we repeatedly do it until we reach a certain threshold.

4.2 Effective Solution

Our goal is to solve the optimization problem of finding critical blocks, see equation (1). To solve this problem, we use a hierarchy way and first we try to find the nodes which cut the whole graph into two parts. We first introduce an entry vector $v = (v_1, v_2, \dots, v_n)^T$, the element v_i in v is defined as [21]:

$$v_i \equiv \begin{cases} \sqrt{|\bar{S}|/|S|} & \text{if } n_i \in S \\ -\sqrt{|S|/|\bar{S}|} & \text{if } n_i \in \bar{S} \end{cases} \quad (2)$$

The vector v has two important properties:

- The first property is that the inner product of v and $\mathbb{1}$ is zero, we can get this property by equation (3):

$$v \cdot \mathbb{1} = \sum_{i=1}^n v_i = \sum_{i \in S} \sqrt{\frac{|\bar{S}|}{|S|}} - \sum_{i \in \bar{S}} \sqrt{\frac{|S|}{|\bar{S}|}} = 0 \quad (3)$$

In other words, the vector v is orthogonal to the const vector $\mathbb{1}$.

- The second property is that the Euclidean norm of v satisfies $\|v\| = \sqrt{n}$. See equation (4).

$$\|v\|^2 = \sum_{i=1}^n v_i^2 = |S| \frac{|\bar{S}|}{|S|} + |\bar{S}| \frac{|S|}{|\bar{S}|} = |\bar{S}| + |S| = n \quad (4)$$

So with the help of Proposition 1, it will be conveniently verified that:

$$v^T Lv = \sum_{i,j} (v_i - v_j)^2 = 2 \sum(N) \sum_{i \in S, j \in \bar{S}} \left(\frac{E(i,j)}{|S|} + \frac{E(i,j)}{|\bar{S}|} \right) \quad (5)$$

In the equation (5), $\sum(N)$ stands for number of nodes, for a given network, it is a constant number. So solving equation (1) can be equivalently to minimize $v^T Lv$. Our goal can be equivalently rewritten as:

$$\min v^T Lv \quad \text{s.t.} \quad v \perp \mathbb{1}, \|v\| = \sqrt{n} \quad (6)$$

Algorithm 1. Finding Critical Blocks

Input:

the social network G;
 the degree bound for nodes' degree θ ;
 the size bound for critical blocks' size ε ;

Output:

the critical blocks and partitioned parts of the whole network;

- 1: For a given network, filter out trivial nodes bounded by θ , leave them as trivial critical blocks;
 - 2: Build up Laplacian Matrix for the network;
 - 3: Compute the Fiedler Vector of the matrix;
 - 4: Using given heuristic algorithms to get critical blocks;
 - 5: Partition the graph based on found critical blocks and repeat the process from line 2 until reach the constraint of ε ;
-

Recall a fact about the Rayleigh quotient [10]: Let M be a real symmetric matrix. Under the constraint that x is orthogonal to the $i-1$ smallest eigenvectors x_1, x_2, \dots, x_{i-1} , the Rayleigh quotient $\frac{x^T M x}{x^T x}$ is minimized by the next smallest eigenvector x_i and its minimum value is the corresponding eigenvalue λ_i .

For a given Laplacian Matrix L, the smallest eigenvalue λ_1 is 0 and its corresponding eigenvector is $\mathbf{1}$ from Proposition 2. From the above equation we can see that vector v is orthogonal to the const vector $\mathbf{1}$. So the second smallest eigenvector corresponding to λ_2 of L gives a solution that we are looking for. In fact, the second smallest eigenvector is also called Fiedler Vector, due to the pioneer work by M. Fiedler. After computing the second smallest eigenvector, we can get our critical blocks using a heuristic algorithm, the nodes which partition the graph usually lie in the middle of the sorted Fiedler Vector, see Fig 1(b).

In Fig 1(b), we mark the Fiedler Vector value for each node of the network in Fig 1(a). The nodes left to the green node have negative value, while nodes right to the green node have positive value, and the green node itself composes a critical block, which divide the whole graph into two parts. Notice that green node has a relatively small Fiedler Vector value(in magnitude) compared with other nodes.

A similar process can go on using the third smallest eigenvector to give optimal partition for the first two divided parts. In fact, we can compute all the eigenvalues and the corresponding eigenvectors, each time partition the former divided parts using the next smallest eigenvector. However, since we use an iterative algorithm to compute eigenvalues and eigenvectors, the accumulate errors is large, and the results may become not so accurate. To avoid the unreliable phenomenon, we can restart the next computing individually.

Our algorithm FCB (Finding Critical Blocks) is summarized in Algorithm 1. Line 1 and line 2 in Algorithm 1 can be executed at the same time, we filter out the trivial critical block nodes at the time we set up Laplacian matrix. After the filtration, the graph may become unconnected, but from Proposition 4, we know

we get a “nature partition”, and it does not affect the last result. In line 4, we proposed three heuristic algorithms to get critical blocks.

- a). finding the nodes corresponding to the median values of eigenvector, usually these nodes compose the critical block;
- b). using signs of the values of eigenvector to partition the network, and the nodes in the cut set compose the critical block;
- c). sorting the values of eigenvector, whereby a “large” gap in the sorted vector can give a partition for the network.

The Complexity Analysis. The most expensive part of our algorithm is to compute some smallest eigenvectors , solving the eigensystem has a complexity of $O(n^3)$, where n is the number of nodes in a graph. As n increases, it will be slow to solve the eigensystem, but our algorithm has the following properties:

1. The graph we faced is sparse. It has been verified that network connection is very sparse [1];
2. We do not need to compute all the eigenvalues and eigenvectors, there are some methods which are efficient to compute only some partial eigenvectors, such as Lanczos method [10];
3. We use an iterative algorithm, and we don't need a high precision result, the number of iterations is usually small to reach a relatively stable state.

Solving the eigensystem is a well studied problem, and lots of algorithms have been proposed, such as Trace Minimization algorithm and Lanczos method. Lanczos method is acknowledged as an excellent way to solve eigensystem of large scale sparse matrix, especially for the some largest and smallest eigenvectors. If matrix A is sparse, and $A \cdot x$ can be computed in $k \times n$ times ($k \ll n$), Lanczos method can guarantee all the eigenvalues can be computed in $O(mn)$ times, and be stored in $O(m)$ storage space, where m depends on many factors, since our matrices are very sparse, $A \cdot x$ can be computed in $k \times n$ time, $k \ll n$. So the complexity of our algorithm is $O(mn)$.

The Limitations of Algorithm 1. Algorithm 1 is not effective for all types of graph, e.g. clique (complete graph) or k-clique. Because for a clique(or k-clique), all the nodes have the same structure, so the values of Fiedler Vector are all equal. But as illustrated in [4, 11, 23], social networks show community structures, and our algorithm can work well for such type graphs.

5 Experimental Evaluation

We evaluate the performance of our algorithm both on synthetic data and real word datasets, Twitter. The experiments are performed on IBM X3500 with double Xeon Quad core CPU, the memory is 16 GB, and the OS we used is 64 bit CentOS. Our program is written in R language, and the package for computing eigenvectors is ARPACK¹, Fig 3 is plotted using pajek².

¹ <http://www.caam.rice.edu/software/ARPACK/>

² <http://pajek.imfm.si/doku.php>

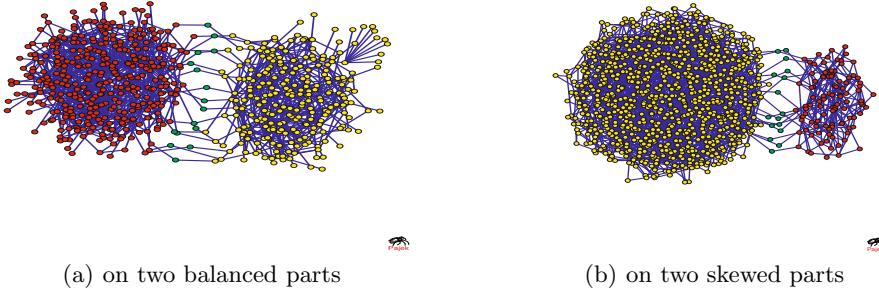


Fig. 3. Experimental results both on balanced and skewed networks

5.1 Experiments on Synthetic Data

The goal of our experiments on synthetic data is to testify the effectiveness, we visualize the graph for small size graph, for it is not so easy to observe critical blocks for large scale graph intuitively. We proceed as follows: first, we generate some types of networks varying from around 800 to 1,000 nodes; then, we run our algorithm for these networks; we plot the results using pajek at last.

Data. We generate two types of networks. The first type is a balanced network, the degree distribution of the network obeys power law distribution. Since social networks have a close-knit form, the network we generated mainly form two parts, that is, there are two “tight” parts and a small number of nodes connect them, and these connecting nodes compose the critical block. The size of two parts are almost equal, they are balanced. See Fig 3(a).

The second type networks we generate are highly skewed. There are two “tight” parts but one part has much more nodes than the other. See Fig 3(b).

Results. We run our algorithm and mark each node as follows: the nodes in critical block is marked with green color, and for the two partitioned parts, we mark them with red and yellow colors separately, for clarity, we do not label the value of Fiedler Vector value for each node in the figure. The experiment results show that the nodes in the critical block found by our algorithm is exactly the connecting nodes we generated, that is, our algorithm is very effective. The experimental results are illustrated in Fig 3.

In Fig 3(a), the data we generated form two parts, one part has 450 nodes, the other has 400 nodes and less than 20 nodes connect them, the two parts are about balanced. In Fig 3(b), the network is highly skewed, which also has two parts, the first part has 900 nodes, and the second part has 100 nodes, less than 20 nodes connect them. We can see that our algorithm is effective for not only balanced network but also for skewed network. For skewed network, our algorithm do not partition the two parts with equal size but in a more reasonable way.

5.2 Experiments on Real World Datasets

We also run our algorithm on real word datasets, Twitter. The data we used is anonymous and there is no privacy problem. The experiments on real world data are mainly to testify the scalability of our algorithm.

Dataset Description. Twitter is a famous micro-blog website, and it is reported that the registered users are almost 500 million, and the number is increasing every day. The data we downloaded contains only a small part of the whole network³. We sample the social networks with about 100,000, 200,000, 500,000, 1,000,000 nodes separately. For every network, we filter out trivial nodes (we filter out the from top 0.1% to top 0.2% trivial nodes, which take away many edges).

Some Measurements. Since there is no objective ground truth to measure the results we found, here we use three measurements.

clustering coefficient [20, 22]. Also called *transitivity*. Clustering coefficient is a measurement of degree to which nodes in a graph tend to cluster together. Evidence suggests that in social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties [23]. And there are two types of clustering coefficient, global clustering coefficient GCC and local clustering coefficient LCC. The GCC is defined as [22]:

$$GCC = \frac{\text{number of closed triplets}}{\text{number of connected triplets of nodes}}$$

Where a triplet consists of three nodes that are connected by either two (open triplet) or three (closed triplet) undirected ties.

The local clustering coefficient LCC_i for a node i is defined as [20]:

$$LCC_i = 2 \frac{|e_{jk}|}{k_i(k_i - 1)} \quad n_i, n_j \in N, e_{jk} \in E$$

The local clustering coefficient of a node in a graph quantifies how close its neighbors are to being a clique (complete graph).

Based on LCC, we also define average LCC for (called ACC) as:

$$ACC = \frac{1}{2} \sum_{i=1}^k LCC_i$$

diameter. The diameter of a graph is the length of the longest geodesic.

number of nodes in critical blocks. Denoted as # in CB.

We run our algorithm on the four datasets only for the first partition, that is, every network is divided into two parts. The Laplacian matrix is stored using sparse matrix format. We record the parameters of # in CB, GCC, ACC, diameter for the original network and for the divided two parts. The results are shown in Table 1.

³ <http://socialcomputing.asu.edu/datasets/Twitter>

Table 1. Experimental Results on Twitter Datasets

Datasets	# in CB	GCC (10^{-4})		ACC		diameter	
		before	after	before	after	before	after
100,000 nodes	143	1.072 1.101	1.099 1.101	0.156 0.169	0.168 0.169	15	10 9
200,000 nodes	198	0.891 0.915	0.912 0.915	0.145 0.157	0.156 0.157	16	11 10
500,000 nodes	279	0.584 0.602	0.601 0.602	0.133 0.143	0.143 0.143	16	11 11
1,000,000 nodes	388	0.362 0.380	0.378 0.380	0.124 0.134	0.132 0.134	18	12 11

Results and Analysis. In Table 1, the GCC column and ACC column and diameter column are split into two columns, “before” column and “after” column, respectively. Because our algorithm partition the network into two parts, the “before” column record the parameters for the original network, and the “after” column record the parameters for the two divided parts. In Table 1, we can see that the number of nodes in critical blocks is relatively small, that means we don’t need affect too many nodes to limit the diffusion over the network. We divide the whole network into two parts, and the GCC and ACC of “before” column represent the original network, and in the “after” column they represent the parameters for two divided parts separately. GCCs are small numbers (in 10^{-4}), for social network is sparse, and filtering out trivial nodes make the network more sparse, and GCC calculates number of closed triplets over number of connected triplets of nodes, for sparse network, it is usually small. We can see both GCC and ACC are improved, that means we divide the network with the “weakest” connected nodes and edges. The diameters show that we have divided the network into two parts, and they are balanced. It should be mentioned that since we have filtered out trivial nodes, our diameters are larger than six, but it does not mean that our social networks disobey “six degree of separation”.

6 Conclusion and Future Work

In this paper, we have investigated the problem of finding critical blocks to limit the diffusion over social networks. We solve such problem as to find critical blocks, then we proved that such a problem is NP-complete and proposed an effective and efficient solution based on spectral theory and analyzed its complexity. The experimental results both on synthetic data and real world datasets showed that there did exist critical blocks connecting different components and our algorithm could find them effectively.

In the future work, we will study on the dynamic social networks. Since the network structure changes all over the time, what we are facing is an evolutionary graph. Another direction we will study is how to implement our algorithm on weighted and directed graph, and how to quantify the weight of two nodes is still a big challenge facing us.

Acknowledgments. This research was supported by the NSFC (Grant No. 61025007, 60933001 and 61100024), National Basic Research Program of China (973, Grant No. 2011CB302200-G) and MOE-Intel Special Fund of Information Technology (MOE-INTEL-2012-06).

References

1. Aggarwal, C.C.: Social Network Data Analytics. Springer Press, Germany (2011)
2. Ahuja, G.: Collaboration networks, structural holes, and innovation: A longitudinal study. *Administrative Science Quarterly* 45, 425–455 (2000)
3. Bapat, R.B.: Graphs and Matrices. Springer, Germany (2011)
4. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. *Science* 286(5439), 509 (1999)
5. Borgatti, S.P., Mehra, A., Brass, D.J., Labianca, G.: Network analysis in the social sciences. *Science* 323(5916), 892–895 (2009)
6. Budak, C., Agrawal, D., Abbadi, A.: Limiting the spread of misinformation in social networks. In: WWW 2011, pp. 497–505. ACM (March 2011)
7. Burt, R.: Structural Holes: The Social Structure of Competition. Harvard University Press, Cambridge (1992)
8. Pinto, P.C., Thiran, P., Vetterli, M.: Locating the Source of Diffusion in Large-Scale Networks. *Physical Review Letters*. Week Ending (August 10, 2012)
9. Fiedler, M.: A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.* 25(100), 619–633 (1975)
10. Golub, G.H., Loan, C.F.V.: Matrix Computations, 3rd edn. Johns Hopkins University Press, USA (1996)
11. Gomez-Rodriguez, M., Leskovec, J., Krause, A.: Inferring networks of diffusion and influence. In: KDD 2010, pp. 1019–1028. ACM (July 2010)
12. Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: A data-based approach to social influence maximization. In: VLDB 2012, pp. 73–84. ACM (2012)
13. Kempe, D., Kleinberg, J.M., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD 2003. ACM (2003)
14. Kleinberg, J., Suri, S., Tardos, E.: Strategic network formation with structural holes. In: EC 2008: ACM Conference on Electronic Commerce. ACM (July 2008)
15. Leskovec, J., Backstrom, L., Kleinberg, J.: Meme-tracking and the dynamics of the news cycle. In: KDD 2007, pp. 497–505. ACM (July 2007)
16. Leskovec, J., Krause, A., Guestrin, C., et al.: Cost-effective outbreak detection in network. In: KDD 2007, pp. 420–429. ACM (2007)
17. Hagen, L., Kahng, A.B.: New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design* 19(2) (September 1992)
18. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS 2001. ACM (March 2001)
19. Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8) (August 2000)
20. Thakar, J., Christensen, C., Albert, R.: Toward understanding the structure and function of cellular interaction networks. In: Handbook of Large-Scale Random Networks. Springer Press, Germany (2008)
21. von Luxburg, U.: A tutorial on spectral clustering. *Stat. and Comp.* 17(4) (2007)
22. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge University Press, England (1994)
23. Watts, D.J., Strogatz, S.: Collective dynamics of small-world networks. *Nature* 393(6684), 440–442 (1998)

Guide Query in Social Networks

Yu-Chieh Lin¹, Philip S. Yu², and Ming-Syan Chen^{1,3}

¹ Dept. of Electrical Engineering, National Taiwan University, Taipei, Taiwan
yuccalin@arbor.ee.ntu.edu.tw

² Dept. of Computer Science, University of Illinois at Chicago, Chicago IL, USA
psyu@uic.edu

³ Research Center for Information Technology Innovation,
Academia Sinica, Taipei, Taiwan
mschen@citi.sinica.edu.tw

Abstract. In this paper, we study a new problem of social guide query, which is to find the most informative neighboring nodes of the query node considering the given requirements. After the target nodes are identified, instead of directly providing information of the target nodes, we would like to find the querier’s friends who are targets or who can introduce the querier to some targets. To answer the social guide query, we define InfScore based on an existing model of influence diffusion. In addition, we define DivScore that can be integrated in the ranking process to appreciate the diversity of possibly accessible target nodes. To evaluate the guide query, we develop a prototype system using the coauthor network of DBLP computer science bibliography. The results show that the guide query is practical, which can provide valuable answers efficiently.

1 Introduction

Recently, online social networks are getting more and more popular. With a huge amount of online social network data, we can analyze social interactions and activities. In this paper, our goal is to find *guides*, informative friends, for a given querier. A guide represents someone who is our friend on the online social network and may provide the information we are looking for. The guide itself may know about the information we need, or the guide may know some other friends who have the information we need. The reason that a guide must be a friend is that we usually get more detailed and helpful information if we ask for information from the persons we have good relationship with.

The motivation of this work can be understood by the following example. When finding a job, a user always wants to know more about the companies she is interested in. If she knows some of her friends are working in those companies, it is not difficult to ask them to get information. However, what if she does not know whether there is any friend working there? Or, what if there is no friend working there? Based on the social graph data, we can find out which friends she should ask first. The suggestion would be the friends who are working in those target companies, or the friends who could introduce some of their friends working there.

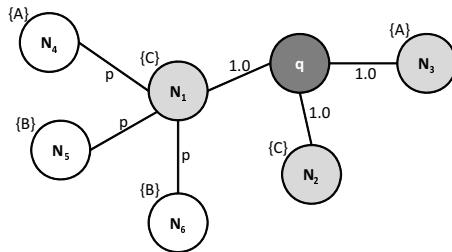


Fig. 1. Example of a part of a social network

The similar scenario can be applied to lots of cases in our life. Although we might find some useful information on the Internet, it is always better to reach someone who has been in the corresponding circumstances. It is worth noting that a querier usually cannot access to information of all other users in the social network. By providing the guides, the social network service provider can help the querier without providing further information of the nodes that are not friends of the querier.

An example social network graph is shown in Figure 1. Node q is the querier, the user who is looking for information. Assumed that q is looking for information A or B , N_3 , N_4 , N_5 , and N_6 are the targets. To access the targets, N_1 and N_3 , who are friends of q , are the guides. N_3 is a guide because it knows A . N_1 is a guide because three of its friends know A or B . N_2 is not a guide because q cannot know A nor B from it. N_3 could be a better guide since q can directly connect to N_3 . However, N_1 could be better since q may access three targets via N_1 . To decide whether N_1 or N_3 is better, we would like to consider edge probability that the information is indeed passed through each edge. The main idea is to use the expected number of influenced target nodes based on some influence diffusion model to represent the informative level for each candidate node, and then to sort the candidate nodes and provide the best few guides as the answer. In this example graph, N_1 is a better guide when $p > 1/3$.

The contribution of this paper is summarized as follows.

- We define the guide query, which is to find the informative neighboring nodes of a given query node.
- We propose a framework to answer the guide query by considering the ability of influence diffusion to target nodes. In addition, the framework can be extended to consider the diversity of possibly accessible target nodes.
- We develop a prototype system using the coauthor network of DBLP computer science bibliography, which can provide valuable answers efficiently.

The rest of the paper is organized as follows. In Section 2, the guide query is defined and the proposed framework for answering the guide query is introduced. Corresponding experiments are performed in Section 3. Related works are introduced in Section 4. The article is concluded in Section 5.

2 Proposed Framework

In this section, we first provide necessary definitions and define the guide query. We introduce the basic framework to answer the guide query based on *InfScore*, and we describe the advanced framework integrating *InfScore* and *DivScore*.

2.1 Problem Definition

In this paper, the social network $G = (V, E)$ is modeled as follows. Each node $N_i \in V$ has some attribute values, which can be matched with a given keyword to decide whether the node is a target node or not. Each edge $e_{i,j} \in E$ is with a probability deciding whether information can be passed through the edge, which depends on the relationship between the two nodes.

With the given social network, we define the guide query as follows.

Definition 1. *Given a query node q and a set of keywords $W = \{w_1, w_2, \dots, w_{|W|}\}$, the guide query is to find the top- k informative neighbors of q considering W .*

The phrase *informative* is the key of the objective. We will design *InfScore* and *DivScore* to decide the informative level in Section 2.2 and Section 2.3 correspondingly. For better understanding, following definitions are provided.

Definition 2. *Given the query node q , we define the neighboring nodes of q as candidate nodes, where C denotes the set of candidate nodes.*

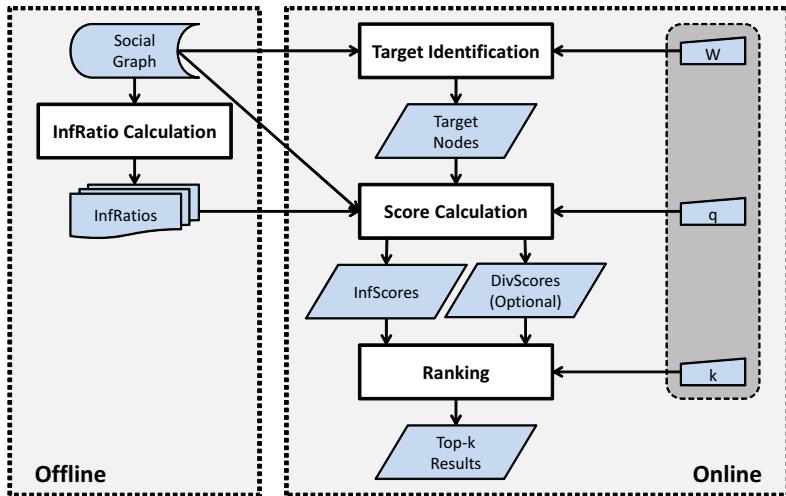
Definition 3. *Given the keyword set W , we define any node having at least one attribute in W as a target node, where T denotes the set of target nodes. Besides, let T_{w_x} denote the set of target nodes having the keyword w_x . Note that $T_{w_x} \subseteq T$, and T is the union of each T_{w_x} .*

Take Fig. 1 for example. Given q and $W = \{A, B\}$, we have $C = \{N_1, N_2, N_3\}$ and $T = \{N_3, N_4, N_5, N_6\}$, where $T_A = \{N_3, N_4\}$ and $T_B = \{N_5, N_6\}$.

2.2 The Basic Framework

In this section, we first define *InfScore*, and we propose the basic framework for answering the guide query based on *InfScore*.

The main idea of *InfScore* is to use the ability of spreading the request to target nodes to represent the informative level for a candidate node. A candidate who can broadly spread the request is able to help the querier access to more target nodes. The IC (Independent Cascade) model [4], which is one of the most popular information diffusion models, is used to define *InfScore*. In the IC model, briefly speaking, every edge is given with a probability, which is identical to the given social graph in the guide query. A node N_i that is influenced at time t will try to influence every neighbor N_j at time $t + 1$, and the edge probability $e(N_i, N_j)$ is the probability that N_j is successfully influenced by N_i . Based on the IC model, we first define *InfRatio* and then use it to define *InfScore*.

**Fig. 2.** Flowchart

Definition 4. Based on the IC model, assuming that N_i is initially influenced, $\text{InfRatio}(N_i, N_j)$ is defined as the probability that N_j is also influenced. Note that N_j can be any node in the social graph.

Definition 5

$$\text{InfScore}(N_i) = \sum_{\forall x} \sum_{\forall N_j \in T_x} \text{InfRatio}(N_i, N_j)$$

As Def. 5, InfScore is an expected weighted sum of the number of target nodes that are influenced when assuming that N_i is initially influenced. The weight is the number of keywords that a target node has. This is to say, if the query node ask for N_i 's help, the number of nodes that are finally accessed is expected to be $\text{InfScore}(N_i)$, where a node having two keywords is counted twice. With the InfScore defined, the guide query in Def. 1 is re-defined as follows.

Definition 6. Given a query node q and a set of keywords $W = \{w_1, w_2, \dots, w_{|W|}\}$, the guide query is to find the neighbors of q with the top- k InfScore considering W .

With Def. 6 in mind, we design the basic framework for answering the guide query as shown in Fig. 2. The Social Graph is first generated based on the given social network dataset, following the descriptions at the beginning of Section 2.1. In the offline phase, InfRatio Calculation calculates and stores InfRatios . In the online phase, Target Identification, Score Calculation, Ranking are performed consecutively according to the input (q, W, k) .

Although InfRatio is calculated in the offline phase, it could be very hard to calculate the exact value of InfRatio since all possible worlds need to be

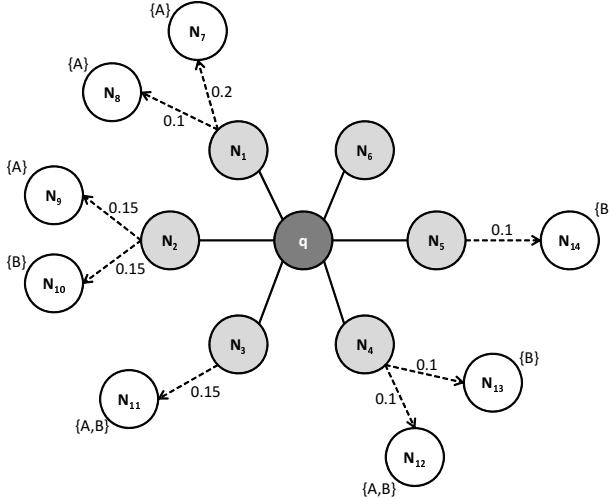


Fig. 3. An example social graph with *InfRatio*

examined, where a possible world is a possible network structure considering whether each edge exists or not. Thus, following previous works using the IC model [1] [4], we apply a large number of simulations to obtain the approximate value of *InfRatios* and store the results for calculating *InfScores* in the online phase. For example, assuming we run 10,000 simulations with N_i initially influenced, and in 4,800 simulations that N_j is finally influenced. We estimate $\text{InfRatio}(N_i, N_j)$ by 0.48, which is a part of *InfScore*(N_i). Details of each process are introduced as follows.

- InfRatio Calculation: In the offline phase, InfRatio Calculation calculates $\text{InfRatio}(N_i, N_j)$ of any (N_i, N_j) pair and then stores $\text{InfRatio}(N_i, N_j)$ of the same N_i as a file.
- Target Identification: The given keywords in W are matched with the attribute values of each node. A node having at least one attribute value equaling to a keyword is identified as a target node. At the end of the process, we obtain T and also T_x of each keyword $w_x \in W$.
- Score Calculation: The set of neighboring nodes of the given query node q is identified as C . For each candidate node in C , *InfScore* is calculated following Def. 5 based on T and *InfRatios*, which are already calculated and stored in the offline phase. At the end of the process, *InfScore* of each candidate node is obtained.
- Ranking: The candidate nodes are sorted by their *InfScores*. The candidate nodes with the top- k *InfScores* are the answer to the guide query.

Example 1. Figure 3 is an example graph for demonstrating the proposed framework. The graph is also used in every example in the rest of this paper. The

dotted line with an arrow from N_i to N_j represents that N_j could be influenced if N_i is initially influenced, and the value besides the dotted line represents the $\text{InfRatio}(N_i, N_j)$ estimated by simulations. With the query node q and keyword set $W = \{A, B\}$, the candidate set C and target set T are decided based on Def. 2 and Def. 3 as follows.

- $C = \{N_1, N_2, N_3, N_4, N_5, N_6\}$.
- $T = \{N_7, N_8, N_9, N_{10}, N_{11}, N_{12}, N_{13}, N_{14}\}$.
- $T_A = \{N_7, N_8, N_9, N_{11}, N_{12}\}$ and $T_B = \{N_{10}, N_{11}, N_{12}, N_{13}, N_{14}\}$.

Following Def. 5, $\text{InfScore}(N_i)$ of each $N_i \in C$ is calculated using the given $\text{InfRatio}(N_i, N_j)$. Take N_2 and N_4 for example.

$$\begin{aligned} \text{InfScore}(N_2) &= \text{InfRatio}(N_2, N_9) + \text{InfRatio}(N_2, N_{10}) \\ &= 0.15 + 0.15 = 0.3 \\ \text{InfScore}(N_4) &= \text{InfRatio}(N_4, N_{12}) \\ &\quad + [\text{InfRatio}(N_4, N_{12}) + \text{InfRatio}(N_4, N_{13})] \\ &= 0.1 + (0.1 + 0.1) = 0.3 \end{aligned}$$

It is worth noting that $\text{InfRatio}(N_4, N_{12})$ is counted twice when aggregating $\text{InfScore}(N_4)$. This is because N_{12} has both keyword A and keyword B in its attributes, which means that $N_{12} \in T_A$ and $N_{12} \in T_B$. Similarly, we have $\text{InfScore}(N_1) = 0.3$, $\text{InfScore}(N_3) = 0.3$, $\text{InfScore}(N_5) = 0.1$, and $\text{InfScore}(N_6) = 0$.

Then we sort the candidate nodes based on their InfScores . Given $k = 2$, the answer to the guide query are two nodes randomly chosen from $\{N_1, N_2, N_3, N_4\}$, which are the nodes with the top InfScore .

2.3 The Advanced Framework

In addition to the InfScore, we define the DivScore as another feature for deciding the informative level. The DivScore is designed to represent the diversity considering the number of possibly accessible target nodes. This is based on the observation that attribute values of a node might not be complete. Since there might be extra information obtained when each target node is accessed, it is better to access more target nodes for more extra information.

With the consideration above in mind, we define the target vector as follows.

Definition 7. For some node N_i , the target vector X_T of size $|T|$ is defined as

$$X_T(N_i)[t] = \frac{\sum_{\forall x | N_j \in T_x} \text{InfRatio}(N_i, N_j)}{\text{InfScore}(N_i)},$$

where $T[t] = N_j$. Each item in X_T is a normalized InfScore value, which describes the probability distribution on different target nodes.

Table 1. Scores and target vectors

N_i	InfScore	X_T	DivScore
N_1	0.3	[0.67, 0.33, 0, 0, 0, 0, 0]	0.915
N_2	0.3	[0, 0, 0.5, 0.5, 0, 0, 0]	1.0
N_3	0.3	[0, 0, 0, 0.1, 0, 0, 0]	0
N_4	0.3	[0, 0, 0, 0, 0.67, 0.33, 0]	0.915
N_5	0.1	[0, 0, 0, 0, 0, 0, 1]	0
N_6	0	[0, 0, 0, 0, 0, 0, 0]	0

Example 2. Referring to the example graph in Fig. 3, we obtain the target vector X_T based on Def. 7. Take N_2 and N_4 for example.

$$X_T(N_2) = [0, 0, \frac{0.15}{0.3}, \frac{0.15}{0.3}, 0, 0, 0, 0] = [0, 0, 0.5, 0.5, 0, 0, 0, 0],$$

$$X_T(N_4) = [0, 0, 0, 0, 0, \frac{0.2}{0.3}, \frac{0.1}{0.3}, 0] = [0, 0, 0, 0, 0, 0.67, 0.33, 0].$$

We then define *DivScore* corresponding to the target vector X_T .

Definition 8

$$\text{DivScore}(N_i) = \sum_{1 \leq t \leq |T|} -\log X_T(N_i)[t] \times X_T(N_i)[t]$$

Example 3. With the target vectors above, based on Def. 8, *DivScore* values are calculated as

$$\text{DivScore}(N_2) = (-\log 0.5 \times 0.5) + (-\log 0.5 \times 0.5) = 1.0,$$

$$\text{DivScore}(N_4) = (-\log 0.67 \times 0.67) + (-\log 0.33 \times 0.33) = 0.915.$$

Referring to the flowchart in Fig. 2, with the additional *DivScore* calculated in Score Calculation, Ranking simultaneously considers *InfScores* and *DivScore* to answer the guide query. The ranking process is designed to appreciate the candidate node which have a good rank based on each score. Instead of assigning the weights and then summing up *InfScore* and *DivScore*, we sort the candidate nodes by each score and obtain two ranking lists. For each candidate node, the worse rank among the two ranking lists is used for the final ranking process. The candidate nodes with the top- k smallest worse rank are the answer to the guide query. For two nodes with the same worse rank, the other rank is compared to decide the final rank.

Example 4. Referring to the example graph in Fig. 3, Table 1 presents *InfScore* and *DivScore* of all candidate nodes and corresponding target vectors. We rank the candidate nodes according to each score and present the results in Table 2. For each candidate node, the right column is the worse rank among its two ranks. Given $k = 2$, the answer to the guide query are the two nodes with the top-2 smallest worse ranks, which are N_2 and either of $\{N_1, N_4\}$.

Table 2. Ranking lists

Rank	InfScore	DivScore	Worse Rank
1	$N_1 N_2 N_3 N_4$	N_2	N_2
2	—	$N_1 N_4$	$N_1 N_4$
3	—	—	—
4	—	$N_3 N_5 N_6$	N_3
5	N_5	—	N_5
6	N_6	—	N_6

3 Experimental Evaluation

We implement the proposed framework to assess its effectiveness and efficiency.

3.1 Implementation

The DBLP dataset [8] is used to construct the social network graph. The framework is implemented on a PC with Intel i7 CPU and 3GB memory. A node is an author, and an edge between two nodes is the coauthor relationship. Referring to previous works using the IC model [1] [4], the probability of each edge is assigned based on the WC (weighted cascade) model. In this model, the probability of edge $e(N_i, N_j)$ is $1/d(N_j)$, where $d(N_j)$ is the in-degree of N_j . The abbreviated conference names of an author's publications are regarded as the attributes of the corresponding node. Also, the given keywords of the guide query are chosen from the abbreviated conference names. In the offline phase, for each node N_i , we perform 10,000 simulations to obtain the $InfRatio(N_i, N_j)$. Besides, we only store the $InfRatio(N_i, N_j) > 0.01$ to save the storage space and the query processing time.

3.2 Experimental Results

We first apply a few guide queries with $q = \text{'Ming-Syan Chen'}$ and $k = 10$. The first query is with $W = [\text{KDD}, \text{SDM}, \text{CIKM}, \text{ICDM}, \text{PKDD}]$, which are top conferences of the data mining field. The resulting top- k ranking lists are shown in Table 3. The left list ranks candidates based on only *InfScore*, which is also the answer of the basic framework. The middle list ranks candidates based on only *DivScore*. The right list ranks candidates based on the worse ranks, and the list is the answer of the advanced framework. Comparing the answers of the basic and the advanced framework, we find that some authors have good ranks in both answers, such as Philip S. Yu and Wang-Chien Lee. These authors are expected to help the querier reach a good number of target authors with a good diversity. On the contrary, there are some authors ranked well only in the basic framework, such as Chang-Hung Lee and Cheng-Ru Lin. Although these authors are expected to help the querier reach a good number of target nodes,

Table 3. Ranking lists considering $W = [\text{KDD}, \text{SDM}, \text{CIKM}, \text{ICDM}, \text{PKDD}]$

Rank	InfScore	DivScore	Final Rank
1	Philip S. Yu	Philip S. Yu	Philip S. Yu
2	Wang-Chien Lee	Wang-Chien Lee	Wang-Chien Lee
3	Wen-Chih Peng	Wen-Chih Peng	Wen-Chih Peng
4	De-Nian Yang	De-Nian Yang	De-Nian Yang
5	Kun-Ta Chuang	Jian Chih Ou	Jen-Wei Huang
6	Chang-Hung Lee	Chih-Ya Shen	Keng-Pei Lin
7	Cheng-Ru Lin	Yi-Hong Chu	Mi-Yen Yeh
8	Jen-Wei Huang	Pin-Chieh Sung	Chih-Hua Tai
9	Jan-Ming Ho	Chi-Yao Tseng	Kun-Ta Chuang
10	Chung-Min Chen	Ying-Ju Chen	Chi-Yao Tseng

Table 4. Time consumption considering $W = [\text{KDD}, \text{SDM}, \text{CIKM}, \text{ICDM}, \text{PKDD}]$

Process	Basic Framework (sec.)	Advanced Framework (sec.)
Target Identification	0.068	0.066
Score Calculation	1.682	2.853
Ranking	0.006	0.022
Total	1.756	2.941

the required information can be provided by only few targets. The results are identical to our expectation.

Table 4 presents the time consumed to answer the guide query using the basic and the advanced frameworks. It is shown that the guide query is answered in acceptable time. It is worth noting that in the scenario that a server is responsible for only a part of nodes in the social graph, *InfRatios* can be loaded to memory in advance and then the processing time can be largely reduced. Comparing the two frameworks, the advanced framework costs more time in both the Score Calculation and the Ranking processes. Extra time is necessary for constructing the target vector and for integrating the linking lists ranked by *InfScore* and *DivScore* to provide the final answer. In each framework, the Score Calculation process costs most time because the pre-calculated *InfRatios* need to be read from files, and *InfRatio* of each N_i, N_t needs to be found for aggregation.

In the following, for the same q and the same k , we apply the guide query considering different W , which are $W_1 = [\text{SIGMOD}, \text{PODS}, \text{VLDB}, \text{ICDE}, \text{ICDT}]$, $W_2 = [\text{KDD}, \text{CIKM}, \text{PKDD}]$, and $W_3 = [\text{KDD}, \text{SDM}, \text{CIKM}, \text{ICDM}, \text{PKDD}]$. W_1 includes top conferences in the database field, W_3 includes top conferences in the data mining field, and W_2 is a subset of W_3 . We apply the guide query using the advanced framework and we present the resulting ranking lists in Table 5. It is observed that for each keyword set, the authors with the top ranks have published papers on the corresponding conferences, or have coauthored papers with other authors who have published papers on the corresponding conferences, or

Table 5. Ranking lists considering different W

Rank	W_1	W_2	W_3
1	Philip S. Yu	Philip S. Yu	Philip S. Yu
2	Wang-Chien Lee	Wang-Chien Lee	Wang-Chien Lee
3	Wen-Chih Peng	Wen-Chih Peng	Wen-Chih Peng
4	Kun-Ta Chuang	De-Nian Yang	De-Nian Yang
5	Chung-Min Chen	Jen-Wei Huang	Jen-Wei Huang
6	Shan-Hung Wu	Kun-Ta Chuang	Keng-Pei Lin
7	Jen-Wei Huang	Chi-Yao Tseng	Mi-Yen Yeh
8	Keng-Pei Lin	Chih-Hua Tai	Chih-Hua Tai
9	De-Nian Yang	Jian Chih Ou	Kun-Ta Chuang
10	Mi-Yen Yeh	Yi-Hong Chu	Chi-Yao Tseng

Table 6. Time consumption considering different W

Process	W_1 (sec.)	W_2 (sec.)	W_3 (sec.)
Target Identification	0.063	0.046	0.066
Score Calculation	1.913	2.308	2.853
Ranking	0.019	0.017	0.022
Total	1.995	2.371	2.941

both. Although the database and data mining fields are closely related, considering W_1 , W_2 , and W_3 results in different ranking lists. Since some authors are in only one list, it is demonstrated that being able to consider different keyword sets is necessary for the guide query.

Table 6 presents the time consumed to answer the guide query considering W_1 , W_2 , and W_3 . Considering W_3 costs more time than considering W_1 . This is because the candidate nodes are more related to the data mining field, where there are more possibly accessible target nodes when considering W_3 . Considering W_3 also costs more time than considering W_2 . Since $W_2 \subset W_3$, considering W_3 implies more possibly accessible target nodes are involved in the Score Calculation process.

To further examine the efficiency, we randomly choose a query node from the nodes with degrees larger than 10 to apply the guide query considering W with different sizes. We first apply the guide query considering a single keyword and then incrementally add a keyword into W . The average consumed time on different size of keyword sets are presented in Fig. 4(a). It is worth noting that even if considering 10 keywords, the guide query is still answered without much processing time. However, when there are more input keywords, more processing time is required. This is because when there are more input keywords, there are also more possibly accessible target nodes, as shown in Fig. 4(b).

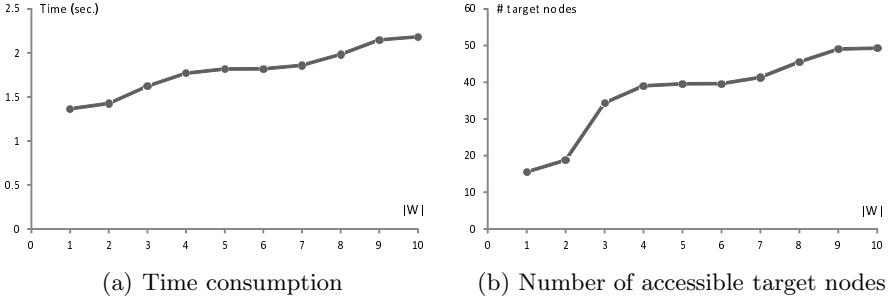


Fig. 4. Results on an incremental keyword set

4 Related Work

With the population of online social network services, there are numerous papers discussing and modeling the online social networks. Some previous works finding the experts in social networks [2] [7]. However, finding the experts is not enough for us. Our goal is to find the most informative neighbors of the query node. Besides, the expert finding algorithm is time-consuming such that it is not suitable for our online target identification process.

Some previous works are on finding gateways or connected parts between the source group and the target group [3] [6] [9] [11]. They are looking for the best group of nodes, which are connected [3] [6] [11] or unconnected [9], to be the bridge from the source group to the target group. Similar to our purpose, they are distinguishing important nodes from others in a social network with full knowledge. However, what we intend to figure out are the informative nodes among the querier's neighboring nodes, but not among all nodes in the graph.

In [11], Wu, Tang, and Bo are finding a tight social graph including the source nodes and the target nodes. The resulting social graph provides insights of guides. However, the resulting social graph does not provide any score of each node such that we cannot rank the querier's neighboring nodes. Another concern is when considering several keywords, the graph might be too large and too complicated for observation.

Many previous works are describing the social network using information diffusion models based on two widely-used fundamentals models [5] [10], which are the independent cascade model (IC) and the linear threshold (LT) model [4]. Some works modify the models to describe different behaviors [5] [10], and some works choose the initially influenced nodes to maximize the final influence [1]. Instead, we propose to use simulated results of the IC model as scores for estimating the ability to help the querier spread the request to target nodes.

5 Conclusion

In this paper, we define the guide query on social network graphs. The guide query is to find the most informative neighbors of the query node considering

given requirements. We define *InfScore* to represent the informative level of a node, based on the popular IC influence diffusion model. We also propose a two-phase framework to answer the guide query efficiently. The offline phase calculates *InfRatio*, which is the main component of *InfScore*, and stores them. The online phase identifies the target nodes, calculates *InfScores*, and ranks the candidate nodes. In addition, we define *DivScore* for extending the framework by considering the diversity of possibly accessible target nodes. We implement the proposed framework for experiments. Example query results are presented to show that the guide query is effective and can be answered efficiently.

References

1. Chen, W., Wang, C., Wang, Y.: Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In: Proc. of 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 1029–1038 (2010)
2. Deng, H., King, I., Lyu, M.R.: Formal Models for Expert Finding on DBLP Bibliography Data. In: Proc. of 8th IEEE Int'l Conf. on Data Mining (ICDM), pp. 163–172 (2008)
3. Faloutsos, C., McCurley, K.S., Tomkins, A.: Fast Discovery of Connection Subgraphs. In: Proc. of 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 118–127 (2004)
4. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the Spread of Influence Through a Social Network. In: Proc. of 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 137–146 (2003)
5. Kimura, M., Saito, K., Nakano, R.: Extracting Influential Nodes for Information Diffusion on a Social Network. In: Proc. of 22nd AAAI Conf. on Artificial Intelligence (AAAI), pp. 1371–1376 (2007)
6. Koren, Y., North, S.C., Volinsky, C.: Measuring and Extracting Proximity in Networks. In: Proc. of 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 245–255 (2006)
7. Lappas, T., Liu, K., Terzi, E.: Finding a Team of Experts in Social Networks. In: Proc. of 15th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 467–476 (2009)
8. Ley, M.: The DBLP (Digital Bibliography and Library Project) Computer Science Bibliography, <http://www.informatik.uni-trier.de/~ley/db/>
9. Tong, H., Papadimitriou, S., Faloutsos, C., Yu, P.S., Eliassi-Rad, T.: BASSET: Scalable Gateway Finder in Large Graphs. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6119, pp. 449–463. Springer, Heidelberg (2010)
10. Wang, Y., Cong, G., Song, G., Xie, K.: Community-based Greedy Algorithm for Mining Top-K Influential Nodes in Mobile Social Networks. In: Proc. of 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD), pp. 1039–1048 (2010)
11. Wu, S., Tang, J., Gao, B.: Instant Social Graph Search. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS, vol. 7302, pp. 256–267. Springer, Heidelberg (2012)

Probabilistic Graph Summarization

Nasrin Hassanlou, Maryam Shoaran, and Alex Thomo

University of Victoria, Victoria, Canada

{hassanlou,maryam,thomo}@cs.uvic.ca

Abstract. We study group-summarization of probabilistic graphs that naturally arise in social networks, semistructured data, and other applications. Our proposed framework groups the nodes and edges of the graph based on a user selected set of node attributes. We present methods to compute useful graph aggregates without the need to create all of the possible graph-instances of the original probabilistic graph. Also, we present an algorithm for graph summarization based on pure relational (SQL) technology. We analyze our algorithm and practically evaluate its scalability using an extended Epinions dataset as well as synthetic datasets. The experimental results show that our algorithm produces compressed summary graphs in reasonable time.

1 Introduction

Graphs are very popular in modeling social networks, protein interactions, web and communication networks, and semistructured data. Nodes in such graphs represent objects or users and edges depict relationships between them. Also, there is often a set of characterizing attributes assigned to each node, such as age, location, function, etc.

As graphs of millions of nodes and their relationships are ubiquitous now, e.g. Facebook, Twitter, Weibo, or DBpedia, there is a pressing need to summarize graphs in order to have a representation that can be consumed by human analysts. In this paper, we consider a graph summarization notion in which nodes are grouped based on node attributes and groups are connected by edges representing inter-group connectedness.

Being able to group graph nodes and edges is only the first step in understanding real graphs. Another challenge is the uncertainty or imprecision of edges, which represent the connectedness or influence of nodes to each other. *Probabilistic* graphs are commonly used to model networks with uncertainties on the relationships between nodes. An important application of probabilistic graphs is in social networks, where the users' influence is modeled as probabilities on the edges [4,5]. Uncertainty can also be a result of data collection processes, machine-learning methods employed in preprocessing, and privacy-preserving processes. Our focus in this work is on graphs where edges (relationships) have existence or influence probabilities as in [4, 5], and we address the problem of summarizing such probabilistic graphs.

Based on the notion of “possible worlds” for probabilistic databases [1–3, 6], a probabilistic graph G defines a set of regular graphs called possible instances.

Assigned to each possible instance there is an existence probability. While the theoretical framework of possible worlds is useful to define what we want, e.g. the mean group connectedness over the possible instances, the number of possible instances is exponential in the size of the original graph, thus rendering approaches that materialize the possible instances very infeasible in practice. Therefore, we present a method that, while based on the possible worlds semantics, does not create any possible instance at all of the original graph. More specifically, we give characterization theorems to compute expected values of the aggregations included in the summary graph using the edge probabilities only.

The massive size of graph data, such as social networks, requires devising effective management methods that employ disk operations and do not necessarily need to load the entire graph in the memory. We present a summarization algorithm that is SQL-based and employs relational operations to create the summary graph. Notably, using relational technology for solving graph problems has been shown to satisfactorily support other graph problems as well (cf. [8, 11, 13]). Experimentally we evaluate our algorithm by implementing it on an Epinions dataset and show that our presented approach is scalable and efficiently computes aggregates on large datasets. In summary, our contributions are:

1. We present a framework for group-based summarization of probabilistic graphs. Our summarization produces useful expected values for the strength of inter-group connectedness.
2. We give characterization theorems for the aggregates of our graph summarization. Some of our results involve sophisticated probabilistic reasoning.
3. We present an algorithm to compute the aggregates of our graph summarization that can be implemented completely using relational operators in an RDBMS. This is a desirable advantage as relational databases are a sound and mature technology that has been proven to scale for very large data.
4. We conduct an experimental evaluation on a real life dataset and synthetic datasets. Our experiments show the scalability of our algorithm in producing summary graphs in reasonable time.

Organization. We review related work in Section 2. In Section 3 we define our method for summarizing regular graphs. In Sections 4 and 5 we define probabilistic graphs and introduce our probabilistic graph summarization method. The theorems and proofs for our probabilistic method are also presented in Section 5. In Section 6 we propose an algorithm to implement our method. In Section 7 we explain the implementation of our algorithm on an Epinions dataset and analyze the efficiency and scalability of our method. Section 8 concludes the paper.

2 Related Work

Summarization of regular (non-probabilistic) graphs has been studied with respect to different aspects (cf. [14–16]). Various problems have been studied on

probabilistic graphs (cf. [7, 9, 10, 12, 17]). However, to the best of our knowledge we are the first to address the problem of summarization of uncertain data graphs.

Grouping the nodes of a graph based on a set of attributes is one of the most common techniques to summarize graphs. Tian et al. [14, 15] introduce a framework to interactively produce such summary graphs. In [16], Zhao et al. introduce graph cubes which are graph summaries created using node grouping operations based on selected attributes.

3 Graph Summarization

We denote a graph database as $G = (V, E)$, where V is the set of nodes, and $E \subseteq V \times V$ is the set of edges connecting the nodes.

Furthermore, there is a set of attributes A_1, A_2, \dots, A_d associated with the nodes. Attributes can be nominal or numerical. Numerical attributes can be discretized as in [15].

We represent the attribute values for a node $v \in V$ as a d -tuple (a_1, a_2, \dots, a_d) , where a_i , for $i \in [1, d]$, is the value of A_i for v .

Let \mathcal{A} be a subset of node attributes. Using \mathcal{A} we group the nodes of G in the usual GROUP BY way and obtain a set $\mathcal{V}_{\mathcal{A}}$ of node groups. Now we have

Definition 1. *The \mathcal{A} -grouping graph is $\mathcal{G}_{\mathcal{A}} = (\mathcal{V}_{\mathcal{A}}, \mathcal{E}_{\mathcal{A}})$ where*

$$\mathcal{E}_{\mathcal{A}} = \{(g', g'') : g', g'' \in \mathcal{V}_{\mathcal{A}} \text{ and } \exists v' \in g' \text{ and } \exists v'' \in g'' \text{ such that } (v', v'') \in E\}.$$

Definition 2. *The \mathcal{A} -graph summarization (A-GS) is a node-edge weighting pair of functions (w_1, w_2) , where*

$$\begin{aligned} w_1 &: \mathcal{V}_{\mathcal{A}} \longrightarrow \mathbb{N} \\ w_2 &: \mathcal{E}_{\mathcal{A}} \longrightarrow \mathbb{N} \times \mathbb{N} \times \mathbb{N} \\ w_1(g) &= |g| \\ w_2(g', g'') &= (x, y, z), \text{ where} \\ x &= |\{v' \in g' : \exists v'' \in g'', \text{ s.t. } (v', v'') \in E\}| \\ z &= |\{v'' \in g'' : \exists v' \in g', \text{ s.t. } (v', v'') \in E\}| \\ y &= |\{(v', v'') : v' \in g', v'' \in g'', (v', v'') \in E\}|. \end{aligned}$$

Fig. 1.(a) shows a graph containing seven nodes. Consider the color of the nodes to be the grouping attribute. Fig. 1.(b) shows the \mathcal{A} -graph summarization of the graph in Fig. 1.(a) with the corresponding values of the w_1 and w_2 measures.

4 Probabilistic Graphs

A probabilistic graph is $G = (V, E)$ (as above), however, associated with each edge e there is a probability $p(e)$ expressing the confidence on the existence of

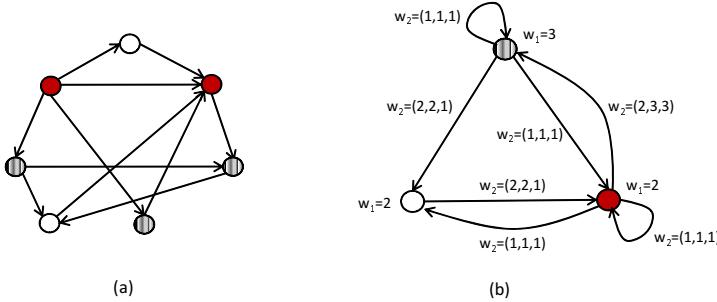


Fig. 1. (a) A Graph G . (b) The summary graph of G .

e. A probabilistic graph defines a set of *possible instances* (PIs). We denote the set of all possible instances of a probabilistic graph G as $\mathcal{PI}(G)$ or \mathcal{PI} if G is clear from the context.

A possible instance (PI) of G is denoted as $PI_i(G)$ or simply PI_i . Each PI is a regular graph derived from the probabilistic graph where each edge either exists or does not. The existence probability of each PI is computed as

$$p(PI) = \prod_{e \in E(PI)} p(e) \cdot \prod_{e \notin E(PI)} (1 - p(e)) \quad (1)$$

where $E(PI)$ is the set of the edges existent in the possible instance PI . Each edge e in the probabilistic graph G appears in a subset of the PIs. For a given edge e the probabilities of PIs containing e sum up to the confidence value (probability) of e , which is denoted as $p(e)$. That is, we have $p(e) = \sum_{E(PI) \ni e} p(PI)$. If e has confidence 1, then it appears in all of the PIs.

Since the number of PIs doubles with each additional probabilistic edge in the input graph, the result of queries on these graphs is exponential in the size of the input graph.

5 Probabilistic Graph Summarization

We define summarization of probabilistic graphs in a similar way as in Definition 2. However, having probabilistic edges between nodes in a graph results in probabilistic edges between groups in the summary graph. Thus, instead of the *exact* value of w_2 the *expected* value should be computed for each of its elements x , y , and z . Note that, the exact value of w_1 is computable as in the non-probabilistic case.

Let g and g' be two groups of nodes in the summary graph \mathcal{G}_A . In each PI the set of edges that connect the nodes of these two groups can be different, and hence, the *exact* values of x , y , and z can differ in the summary graph, corresponding to each PI. The expected values for x , y , and z in \mathcal{G}_A can be

computed using the basic formula for the expected value of random variables. For example, for the expected value of x we have $E[X] = \sum_{PI} x_i \cdot p(PI_i)$, where X is the random variable representing the x measure. Note that, using this formula directly requires building all the possible instances of the original graph.

In the following we present (and prove) equations that compute $E[X]$, $E[Y]$, and $E[Z]$ by using only the probability of edges in G with no need to create all PIs.

Proposition 1. *For any subgraph G' of a probabilistic graph G we have $\sum_{PI \in \mathcal{PI}(G')} p(PI) = 1$.*

Theorem 1. *Let g and g' be two groups in a probabilistic summary graph G , and let $E_{v_j} = \{e_1, \dots, e_{n_j}\}$ be the set of edges connecting a node v_j in g to the nodes of g' . We have that*

$$E[X(g, g')] = E[X] = \sum_{v_j \in g} \left(1 - \prod_{e \in E_{v_j}} (1 - p(e)) \right).$$

Proof. Let $W = \{v_1, \dots, v_{|W|}\}$ be the set of nodes in group g which are connected to the nodes of group g' in G , and let $W_{PI} \subseteq W$ be the set of nodes in group g which are connected to the nodes of group g' in the possible instance PI of G . Also, let $m = |\mathcal{PI}(G)|$. We have that

$$\begin{aligned} E[X] &= \sum_{PI_i \in \mathcal{PI}(G)} x_i \cdot p(PI_i) = \underbrace{p(PI_1) + \dots + p(PI_1)}_{x_1 \text{ times}} + \dots \\ &\quad + \underbrace{p(PI_m) + \dots + p(PI_m)}_{x_m \text{ times}} \end{aligned}$$

where x_i is the number of nodes in g that are connected to some nodes of g' in the instance PI_i . That is, $x_i = |W_{PI_i}|$.

We can organize this equation in a different way. Note that for each node v_j , the term $p(PI_i)$ appears once in the right hand summation if $v_j \in W_{PI_i}$. Therefore, we can rewrite the equation as

$$E[X] = \sum_{W_{PI} \ni v_j} p(PI) + \dots + \sum_{W_{PI} \ni v_{|W|}} p(PI). \quad (2)$$

Now we compute the value of each term above. From equality $\sum_{PI \in \mathcal{PI}} p(PI) = 1$ we have that

$$\sum_{W_{PI} \ni v_j} p(PI) + \sum_{W_{PI} \not\ni v_j} p(PI) = 1. \quad (3)$$

As defined, $E_{v_j} = \{e_1, \dots, e_{n_j}\}$ is the set of edges incident to v_j which connect v_j to some nodes in g' . The first sum in (3) includes possible instances where at

least one of the edges in E_{v_j} exists. The second sum includes possible instances where none of the edges in E_{v_j} exists.

Now, suppose G' is a probabilistic graph constructed from G by removing all the edges in E_{v_j} . That is, the probability of existence of those edges is zero in G' . Since each possible instance of G can be constructed from G' and based on (1), we can rewrite Equation (3) as

$$\sum_{PI \in \mathcal{PI}(G')} p(PI(G')) \cdot \sum_{S \in 2^{E_{v_j}}, S \neq \emptyset} \left(\prod_{e \in S} p(e) \cdot \prod_{e \in S^c} (1 - p(e)) \right) + \\ \sum_{PI \in \mathcal{PI}(G')} p(PI(G')) \cdot \prod_{e \in E_{v_j}} (1 - p(e)) = 1$$

where $\mathcal{PI}(G')$ is the set of all possible instances of graph G' , and S is a set in the power set of E_{v_j} . Since $\sum_{PI \in \mathcal{PI}(G')} p(PI) = 1$ (Proposition 1), we have that

$$\sum_{W_{PI} \ni v_j} p(PI(G)) = \sum_{PI \in \mathcal{PI}(G')} p(PI(G')) \cdot \sum_{S \in 2^{E_{v_j}}, S \neq \emptyset} \left(\prod_{e \in S} p(e) \cdot \prod_{e \in S^c} (1 - p(e)) \right) \\ = 1 - \prod_{e \in E_{v_j}} (1 - p(e)) \quad (4)$$

and using Equations (2) and (4) we have

$$E[X] = \sum_{W_{PI} \ni v_1} p(PI) + \dots + \sum_{W_{PI} \ni v_{|W|}} p(PI) = \sum_{v_j \in W} \left(1 - \prod_{e \in E_{v_j}} (1 - p(e)) \right).$$

This proves the theorem. \square

For the expected value of y we present the following theorem.

Theorem 2. *In the summary graph, the expected value for y , $E[Y]$, is the sum of the probabilities of the edges going from one group to the other.*

Proof. Let $m = |\mathcal{PI}(G)|$ and let $S = \{e_1, \dots, e_{|S|}\}$ be the set of all probabilistic edges (with non-zero probability) that connect the nodes of two given groups in a probabilistic summary graph. Let also $E(PI_i)$ be the set of edges in an instance PI_i . We have that

$$E[Y] = \sum_{PI_i \in \mathcal{PI}(G)} y_i \cdot p(PI_i) = \underbrace{p(PI_1) + \dots + p(PI_1)}_{y_1 \text{ times}} + \dots \\ + \underbrace{p(PI_m) + \dots + p(PI_m)}_{y_m \text{ times}}$$

where y_i is the number of edges in S that exist in PI_i . Now, we can organize this equation in a different way. Note that for each edge $e_j \in S$, if $e_j \in E(PI_i)$, the term $p(PI_i)$ appears once in the right hand summation. Therefore, we can rewrite the equation as

$$E[Y] = \sum_{E(PI_i) \ni e_1} p(PI_i) + \cdots + \sum_{E(PI_i) \ni e_{|S|}} p(PI_i).$$

On the other hand, for each edge e we have that $p(e) = \sum_{E(PI_i) \ni e} p(PI_i)$. Thus, $E[Y] = p(e_1) + \cdots + p(e_{|S|}) = \sum_{e \in S} p(e)$, and this proves the theorem. \square

6 Algorithm

In this section we present our algorithm to build the summary graph of a probabilistic graph. We assume that the probabilistic graph is stored in database tables. The first primary table is the *Nodes* table which consists of all the nodes in the graph and their attribute values. The second is the *Edges* table which stores all the node connections (edges) in the graph. We assume that each edge has an existence probability which is stored in the same table as a separate column.

The algorithm starts by grouping the nodes based on the desired attributes. Grouping can start by sorting nodes according to their values on the selected attributes. Then, computing the $E[X]$, $E[Y]$, and $E[Z]$ elements of the w_2 measure for group pairs can be done by using the theorems and formulas provided in Section 5.

nId	A_1	\dots	A_d
1	a_{11}	\dots	a_{1d}
2	a_{21}	\dots	a_{2d}
\vdots			
n	a_{n1}	\dots	a_{nd}

nId1	nId2	prob
1	2	p_{12}
2	1	p_{21}
\vdots		
i	j	p_{ij}

gId1	gId2	E[X]	E[Y]	E[Z]
g_1	g_2	x_{12}	y_{12}	z_{12}
g_2	g_1	x_{21}	y_{21}	z_{21}
\vdots				
g_i	g_j	x_{ij}	y_{ij}	z_{ij}

Fig. 2. Table *Nodes* (left), Table *Edges* (middle), Table *Summary*(right)

The following algorithm uses the *Nodes* and *Edges* tables illustrated in Fig. 2 (two left tables) and returns the w_2 measure in the *Summary* table depicted in Fig 2(right table). All the steps of our algorithm can be expressed in SQL. Due to space constraint we only give the plain language description of the steps here and refer the reader to the full version¹ of the paper for the SQL statements.

¹ <http://webhome.cs.uvic.ca/~maryam/probgraphsum.pdf>

Algorithm 1

Input:

1. Table *Nodes* containing the nodes and their attribute values.
2. Table *Edges* containing the edges with their existence probabilities.
3. Grouping attribute set \mathcal{A} , which is a subset of node attributes.

Output: Table *Summary* consisting of all possible pairs of groups and their expected measures $E[X]$, $E[Y]$, and $E[Z]$.

Method:

1. Assign a group identifier, gId , to each node in the *Nodes* table based on the user selected attributes.
2. Update table *Edges* and add two new columns called $gId1$ and $gId2$. Then, for each record insert the corresponding group IDs of node 1 ($nId1$) and node 2 ($nId2$) into $gId1$ and $gId2$, respectively.
3. Group records in *Edges* based on $nId1$, $gId1$, and $gId2$ using the product of $(1 - prob)$ as the aggregation function, then, insert the result into a temporary table called $K1$ with the aggregate field as *product*.
4. Group records in *Edges* based on $nId2$, $gId1$, and $gID2$ using the product of $(1 - prob)$ as the aggregation function, then, insert the result into a temporary table called $K2$ with the aggregate field as *product*.
5. To compute element $E[X]$ in the w_2 measure, group records in $K1$ based on $gId1$ and $gId2$ using sum of $(1 - product)$ as the aggregation function and store the result in table *Summary*.
6. To compute element $E[Z]$ in the w_2 measure, group records in $K2$ based on $gId1$ and $gId2$ and sum of $(1 - product)$ as the aggregation function and update table *Summary*.
7. To compute element $E[Y]$ in the w_2 measure, sum up *prob* values from table *Edges* by grouping records based on $gId1$ and $gId2$ and update table *Summary*.
8. Return the *Summary* table.

7 Evaluation

In this section we describe the implementation of our algorithm on a real dataset and evaluate its efficiency. We then analyze the scalability of our algorithm by implementing it on synthetic data.

7.1 Dataset

The real dataset we use for the evaluation is a trust network dataset from *Epinions*². Epinions is a website in which users write reviews for different products of different categories or subjects and express trust to each other.

Two different versions of the Epinions dataset are available in the Trustlet website (www.trustlet.org). In this paper we use the *Extended Epinions* dataset.

² <http://www.trustlet.org/wiki/Epinions>.

The ratings in this dataset are about reviews, also called articles. That is, the ratings represent how much a user rates a given article written by another user. This dataset contains about:

- 132,000 users,
- 841,000 statements (trusts and distrusts),
- 85,000 users received at least one statement,
- 1,500,000 articles.

In this dataset, we are interested in finding the strength of the connections between users grouped by the subject of the articles they have written. Using the *users* information and the *statements* we created tables *Nodes* and *Edges*, respectively. In order to have edge existence probabilities, we added the field *prob* in the *Edges* table and filled it with random numbers between 0 and 1 for each record.

7.2 Implementation of Algorithm 1

Since the *Nodes* table created from the Epinions dataset contains only one attribute, *SubjectId*, we use it as the grouping attribute and group Id will be the *SubjectId* (see Step 1 of Algorithm 1).

To assign the *subjectIds* to the nodes in the *Edges* table (Step 2 of Algorithm 1), we join tables *Nodes* and *Edges* twice, once on *userId1* and the second time on *userId2*. The result table called *Joint* represents all the valid edges in the trust graph. After these joins we end up with much more records in the *Joint* table than table *Edges*. The reason is that in the Epinions dataset a user/author may have articles in different subjects. Before joining the tables, we can follow two different strategies.

1. We can consider each distinct *userId-subjectId* pair in *Nodes* table as a node in the graph. In such a graph, we also need to consider the trust between the nodes having identical *userIds*. With the assumption that each user trusts completely on his/herself, we connect all the nodes having the same *userId* to each other with the probability of 1 and add the corresponding records in the *Edges* table. The result graph is very large with billions of nodes and edges. Fig. 3 depicts this strategy to build the desired graph from the available dataset.
2. We can consider just one subject for each user and remove the other records for that user from the *Nodes* table. In this approach, there will be one node for each user in the graph. Applying this strategy we built a graph consisting of 130,068 nodes each corresponding to a record in *Nodes* table, and 785,286 edges corresponding to the records in the *Joint* table. The number of distinct subjects (groups) was 11,224. This graph is large enough and can be useful for evaluating our algorithm as well.

We have followed both strategies for our evaluation. We performed all the experiments on a machine with Linux server, 12 GB memory, and 3.4 GHz CPU. All steps have been implemented as SQL queries. We executed our queries on MySQL version 5.5.24. In the following section we analyze the results of our experiments on graphs with different sizes.

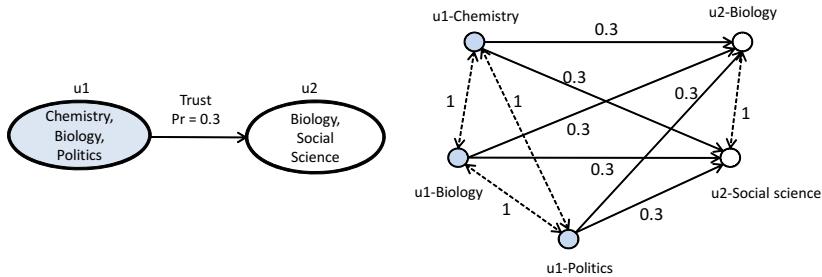


Fig. 3. Graph generation strategy

7.3 Analysis

In this section we analyze the complexity , the efficiency, and the scalability of our algorithm based on the experimental results obtained in the previous section.

7.4 Complexity of the Algorithm

In the first step, a sorting or hashing can be performed to group by the nodes based on their attribute values (the value of *subjectId*). The rest of the algorithm can be completed by scanning the edges in two passes to compute the $E[X]$, $E[Y]$ and $E[Z]$ values.

Considering the memory space, our algorithm can keep the statistics variables for all the groups in the memory. If there is not enough memory, only the information about the groups for which the expected values are requested are kept in the memory. The algorithm can even run in a memory of size equal to the space needed to store statistics variables for only a pair of groups. This is because the algorithm can work with just two groups at a time and compute the expected values of the statistics. However, in this case we would need one pass for each pair of groups.

7.5 Efficiency of the Algorithm

We ran the algorithm on two graphs with different sizes created from the Epinions dataset. The first graph had 840,971 nodes and 103,419,023 edges resulting from restricting the number of subjects to 85, which is almost 0.1% of the whole number of different subjects. This is a reasonable number of groups for a human analyst to easily use in order to understand the interconnectedness of his/her subjects of interest in terms of user trust relationships. The execution time was only 113 seconds, and the produced summary graph contained 85 different groups and 1,691 edges. The second graph was generated using the second strategy illustrated in Section 7.2, which resulted in a graph containing 11,224 different subjects in total. The execution time was only 3.86 seconds, and the produced summary graph contained 11,224 different groups and 35,259 edges.

Basic Graph Statistics			Summary Graph Statistics			
No. Edges	No. Nodes	No. Subjects	No. Edges	No. Nodes	Time (Seconds)	Compression Degree
103,419,023	840,971	85	1,691	85	112.82	99.99%
785,286	130,068	11,224	35,259	11,224	3.86	95.51%

Fig. 4. The experimental results on the Epinions dataset

7.6 Scalability Experiments

In order to analyze the scalability of the algorithm we took advantage of synthetic graphs created based on the trust network structure of the Epinions data. We generated random graphs of different sizes and different number of groups. Each time we simply assigned random group identifiers to each node of the original graph. The experimental results on the datasets having different number of subjects or different graph sizes are shown in Fig. 5.

The left figure in Fig. 5 illustrates the execution time of the summarization algorithm (in seconds) as a function of the number of different groups (subjects) in a graph having 10,000,000 edges. The figure shows that when the graph size is constant, depending on how we group the nodes and how many different groups we get, the execution time can change. The result shows that as the number of different groups increases, the execution time would increase as well in an almost linear manner. Therefore, we can handle the summarization of graphs with large number of groups in reasonable time.

The right figure in Fig. 5 shows the execution time of the algorithm on some graphs of different sizes. In this experiment we group the nodes into exactly 300 different categories each time. The result shows that in the case of constant number of groups, the execution time increases almost linearly based on the graph size. This result shows the scalability of our algorithm.

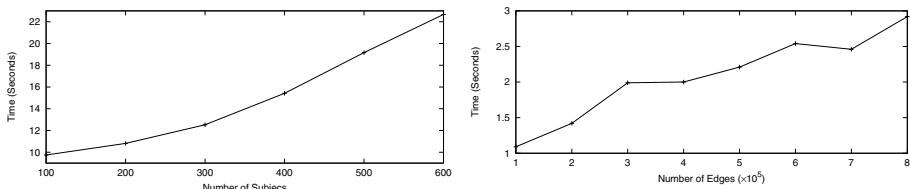


Fig. 5. Left: Execution time vs. number of subjects, Right: Execution time vs. graph size (number of edges)

8 Conclusions

This paper addressed the problem of summarizing probabilistic graphs using a relational database approach. We focused on a useful summarization method which groups the nodes based on a subset of attributes. In the summary graph we considered aggregates which reveal significant information about the groups and the connections between them. We gave theorems to compute these aggregates without the need to compute all possible data graphs from the original probabilistic graph. We also presented an algorithm, which uses pure SQL queries to build the summary graph. We evaluated the proposed algorithm on Epinions data and some synthetic datasets. The evaluation shows that our algorithm is practically scalable to large graphs.

References

1. Abiteboul, S., Grahne, G.: Update semantics for incomplete databases. In: VLDB, pp. 1–12 (1985)
2. Abiteboul, S., Kanellakis, P.C., Grahne, G.: On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.* 78(1), 158–187 (1991)
3. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Widom, J.: Uldbs: Databases with uncertainty and lineage. In: VLDB, pp. 953–964 (2006)
4. Budak, C., Agrawal, D., Abbadi, A.E.: Limiting the spread of misinformation in social networks. In: WWW, pp. 665–674 (2011)
5. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: KDD, pp. 199–208 (2009)
6. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. *VLDB J.* 16(4), 523–544 (2007)
7. Frank, E.H.: Shortest paths in probabilistic graphs 17, 583–599 (1969)
8. Gao, J., Jin, R., Zhou, J., Yu, J.X., Jiang, X., Wang, T.: Relational approach for shortest path discovery over large graphs. *CoRR*, abs/1201.0232 (2012)
9. Pfeiffer III, J.J., Neville, J.: Methods to determine node centrality and clustering in graphs with uncertain structure. In: ICWSM (2011)
10. Kollios, G., Potamias, M., Terzi, E.: Clustering large probabilistic graphs. In: IEEE TKDE (2010)
11. Mayfield, C., Neville, J., Prabhakar, S.: Eracer: a database approach for statistical inference and data cleaning. In: SIGMOD Conference, pp. 75–86 (2010)
12. Potamias, M., Bonchi, F., Gionis, A., Kollios, G.: k-nearest neighbors in uncertain graphs. *PVLDB* 3(1), 997–1008 (2010)
13. Srihari, S., Chandrashekhar, S., Parthasarathy, S.: A framework for SQL-based mining of large graphs on relational databases. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part II. LNCS, vol. 6119, pp. 160–167. Springer, Heidelberg (2010)
14. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: SIGMOD Conference, pp. 567–580 (2008)
15. Zhang, N., Tian, Y., Patel, J.M.: Discovery-driven graph summarization. In: ICDE, pp. 880–891 (2010)
16. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and olap multidimensional networks. In: SIGMOD Conference, pp. 853–864 (2011)
17. Zou, Z., Gao, H., Li, J.: Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In: KDD, pp. 633–642 (2010)

Blind Chance: On Potential Trust Friends Query in Mobile Social Networks

Jinzeng Zhang and Xiaofeng Meng^{*}

School of Information, Renmin University of China, Beijing, China
`{zajize, xfwmeng}@ruc.edu.cn`

Abstract. POTENTIAL-TRUST-FRIENDS-QUERY is an important query in mobile social network, as it enables users to discover and interact with others happen to be in their physical vicinity. In our context, we attempt to find *top-k* mobile users for such query. We propose a novel trust scoring model that encompasses profile similarity, social closeness and interest similarity. Moreover, we devise a current-user-history-record (*CUHR*) index structure to support dynamic updates and efficient query processing. Based on *CUHR* index, we propose a query processing algorithm that exploits candidate generation-and-verification framework to answer queries. Extensive experiments was conducted on the real data set to illustrate the efficiency of our methods.

Keywords: mobile social network, potential friends, trust, check-in history.

1 Introduction

With the rapid development of mobile devices, geo-positioning technologies and social network, location-based services tend to be socialized, which promote the emergence of mobile social network services (MSNS), such as Foursquare, Facebook places, etc. An essential capability of MSNS is to allow mobile users to discover and interact with potential friends who happen to be in their vicinity. There are many important applications in the real world. For example, making friends with other people nearby, seeking assistance in emergency, co-shopping with others with similar demands, taxi-sharing in rush hour, etc.

As we know, trust is critical determinant to share information and establish new relationship between users in daily life because it can protect users from malicious attack and fraud. To support the aforementioned service, we propose a new type of query, called top-k potential trust friends (*PTF*) query, that is to find the top-k high trust mobile users who have common interests and demands with the querying user. The potential friends returned by *PTF* query should satisfy following conditions: (1) they should be annotated with as many terms specified in the query as possible, (2)they should be as close to the querying user as possible in temporal and spatial dimensions, (3) the trust

* This research was partially supported by the grants from the Natural Science Foundation of China (No. 61070055, 91024032, 91124001);the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University(No. 11XNL010); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204).

score should be as high as possible. We introduced a novel trust scoring model by combining three factors: social closeness, interest similarity and profile similarity. To the best of our knowledge, none of former literatures did it like this.

Since continuously moving users change dynamically, it causes dynamic updates of check-in histories. Therefore, it is challenging to retrieve exact query results in a timely fashion. Previous studies on friend discovery just adopt instant monitoring to detect the mobile users in the friends list, which cannot support PTF query [1, 2]. Therefore, we devise an efficient current users-historical records (*CUHR*) index structure to handle this challenge. Based on *CUHR* index, we exploit a candidate generation-and-verification framework to process the *PTF* query. In candidate generation phase, our method not only identifies the current users in the vicinity of query users over the *CU* index, but also searches for the matching users and the high trustworthy users by traversing the *HR* index in parallel. Extensive experiments show that our algorithm achieves high performance.

2 Problem Definition

A potential trust friends (*PTF*) query is denoted as $Q = \langle u_q, q_t, p, \psi \rangle$, where u_q is the identifier of a query user, p and q_t is the current location and time of u_q , ψ is a set of keywords expressing query demands. In order to find top- k mobile users, we have to rank the correlation between Q and these users. Before the further discussion of *PTF* query, several preliminary definitions should be given as follows.

Definition 1 (Check-in Record). A check-in record cr indicates a user id visited a place loc and posted the tweets \mathcal{T} at the timestamp ts , denoted as $\langle id, ts, loc, \mathcal{T} \rangle$.

Definition 2 (Check-in History). A check-in history CH of a user u is a sequence of past check-in records within a time interval $[st, et]$ ($et \geq st$), $CH_u = \{cr | cr_i.id = u \text{ and } cr_i.ts < cr_{i+1}.ts\}$.

Definition 3 (Cover Distance). If the keyword covering area CA of a user u matches the query keywords, the cover distance between Q and u is measured by maximal distance between CA and query Q and denoted as $Cdist(Q, u) = maxdist(Q, CA)$.

A user u can be formally represented as $\langle id, \mathcal{P}, \mathcal{F}, CH \rangle$, where id is the identifier of u , \mathcal{P} is u 's profile information, \mathcal{F} is the friend list. The correlation of Q and u is evaluated with following function, which take account of three essential components, i.e. current spatio-temporal proximity, the spatial-text similarity of check-in histories $sim_{ST}(\cdot)$, and trust score $TS(\cdot)$ between Q and u .

$$aScore(Q, u) = \alpha \frac{\lambda^{q_t - u.t_a}}{dist(Q.p, u.loc_c)} + \beta sim_{ST}(Q, u) + (1 - \alpha - \beta) TS(Q, u) \quad (1)$$

$$sim_{ST}(Q, u) = sim_T(Q, u) / (Cdist(Q.p, u.CA) + \varepsilon_0) \quad (2)$$

In equation (1), $\alpha, \beta \in [0, 1]$, which are used to balance each component's influence on $aScore(Q, u)$; $\lambda \in [0, 1]$ is a time decaying factor. $sim_{ST}(\cdot)$ is determined by the text similarity sim_T and the cover distance $Cdist(\cdot)$ between Q and u . With this metric, PTF query can be formalised as follows.

Definition 4 (POTENTIAL-TRUST-FRIENDS-QUERY, PTF). Given a user set U , PTF query $Q = \langle q_id, q_t, p, \psi, \rangle$ is to find top-k active users $R = \{u_1, u_2, \dots, u_k\}$, such that there does not exist $u' \notin R$ that satisfies $aScore(Q, u') > aScore(Q, u_i)$ where $u_i \in R$.

2.1 Trust Scoring Model

In our trust scoring model, it encompasses three key factors: social closeness, interests similarity and profile similarity.

Social Closeness. Friendship between two users in the social graph can be a good trust estimation metric. The more common friends between users, the more they trust each other. Therefore, the social closeness between query user u_q and mobile user v should be quantified by friendship similarity. We exploit the Dice coefficient to compute the social closeness $SC(u_q, v)$. It is defined as

$$SC(u_q, v) = \frac{2 \times |\mathcal{F}_{u_q} \cap \mathcal{F}_v|}{|\mathcal{F}_{u_q}| + |\mathcal{F}_v|} \quad (3)$$

Where \mathcal{F}_{u_q} and \mathcal{F}_v are the friends set of u_q and v , respectively.

Interest Similarity. People tend to trust others with similar interests or experiences. The historical check-ins provide rich information about a user's interests and hints about when and where a user would like to go. We utilize the spatio-temporal co-occurrence to estimate the interest similarity. We firstly employ a logarithmic scale titled time frame model [5], to split the time interval of check-in histories. Then we use the distance between check-in locations of two users in each time frame to measure interest similarity.

$$IS(u_q, v) = \sum_{i=1}^{maxTf} \sum_{LS_i} \frac{1}{dist(loc_{u_q}, loc_v)} \quad (4)$$

Where $maxTf$ is the maximal time frame among users, LS_i is the union of all locations of u_q and v in the i th time frame.

Profile Similarity. To measure the similarity between users, we compare their profiles. The profile vector $\mathcal{P} = (a_1, a_2, \dots, a_m)$ ($0 < a_m < 1$) of a user is a collection of attributes. For a query user u_q and a mobile user v , we adopt the cosine measure as the profile similarity, which is defined as follows.

$$PS(u_q, v) = \frac{\mathcal{P}_{u_q} \cdot \mathcal{P}_v}{\|\mathcal{P}_{u_q}\| \cdot \|\mathcal{P}_v\|} \quad (5)$$

where \mathcal{P}_u and \mathcal{P}_v is the profile vector of user u_q and v .

Trust Score for Queries. Finally, the trust score of a user v , relative to the PTF query user u_q is obtained by the linear combination of the above three factors.

$$TS(u_q, v) = \lambda \cdot SC(u_q, v) + \mu \cdot IS(u_q, v) + (1 - \lambda - \mu) \cdot PS(u_q, v) \quad (6)$$

where $\lambda, \mu \in [0, 1]$ are used to adjust the tradeoff among three factors.

3 CUHR Index Structure

To facilitate PTF query processing, we design a current users-historical records (CUHR) index. CUHR consists of two ingredients, as shown in Fig. 1, CU memory index (Fig.

1(a)) and HR disk index (Fig. 1(b)). CU is used to index the incoming users and latest check-in records, and HR is responsible for indexing the check-in histories of all users.

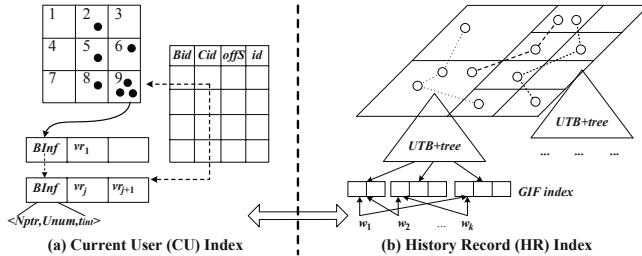


Fig. 1. CUHR Index Structure

In CU index, a regular grid is exploited to index the latest users' visit records. Each cell associates with a linked list of buckets, which contain the meta-data and the latest visit records of each user in ascending of arriving time. The meta-data contains 3 items: a pointer to the next bucket, $Nptr$; the number of stored current users, $Unum$; the visiting time interval of users in this bucket, t_{int} . In mobile scenario, users are dynamically changing. In order to reduce the costs caused by the frequent updates, a hash table is applied to index users on their id . When a new visit record vr of u arrives, we check whether the new cell that vr belongs to is the same as the current cell of u . If not, vr is inserted into the first bucket of the new cell and the corresponding entry in the hash table is updated. Meanwhile, the old and new check-in records are packed together and migrated into disk.

The HR index exploits an adaptive multi-level grid to partition the entire space. Each user's check-in history can be partitioned into a set of sub-sequences, where each is enclosed into the corresponding cell. In order to parallelize PTF query, a $UTB+$ -tree and a grid-inverted file (GIF) index is implemented in each cell. In $UTB+$ -tree, the check-in sub-sequences are organized by the user id and time interval. The GIF index is used to index the text information of check-in sub-sequences in each cell. When update occurs, a block from CU index is inserted into the historical data files. Accordingly, the $UTB+$ -tree and GIF of corresponding cell should be updated.

4 Query Processing for PTF Query

To answer the PTF query, a simple method called CUFinder is to retrieve the current trust users in the vicinity based on the CU index. It exploits an incremental enlargement method to implement range query iteratively, then offer up-to-date top- k results. In contrast to its fast query response time, CUFinder is inefficient for identifying more relevant results. For this purpose, we propose CanGV algorithm, which is based on a candidate generation-and-verification framework.

4.1 Candidate Generation

A *PTF* query involves three tasks: (1) querying top- k current users with largest spatio-temporal proximity, (2) querying top- k users with largest spatial-text similarity, q_2 , (3) querying top- k users with highest trust score, q_3 . We denote the whole query as $Q' = \{q_1, q_2, q_3\}$. Each task q_i ($1 \leq i \leq 3$) of Q' is given a label to describe its priority, q_1 with $l_{q_1} = \frac{\lambda u - u_{ta}}{dist(Q, p, u, loc_c)}$, q_2 with $l_{q_2} = sim_{TS}(Q, u)$ and q_3 with $l_{q_3} = TS(Q, u)$. For each task q_i , we maintain an individual priority heap PH_i to keep track of the users with $\langle u, l_{q_i} \rangle$ and the users with the maximum l_{q_i} will be put on the top of the PH_i . To obtain the candidates efficiently, we utilize a global priority heap GH to keep track of the maximal entries from each PH_i . At each time, the entry that matches q_i is popped from the top of GH and kept in a candidate set CS . Meantime, another entry is popped from corresponding heap PH_i and pushed into GH . Repeating above procedure until the CS has already contained top- k fully matching users. Next, we focus on the processing of task (2) while task(1) and (3) can be solved by adjusting existing algorithms in [3].

Top-k users retrieval with largest $sim_{ST}(Q, u)$. For the task q_2 , we exploit an incremental expansion algorithm to retrieve top- k users that match query keywords and have the minimum cover distance on HR index. In query processing, we utilize a priority heap PH_2 to keep track of top- k users.

At first, we implement a range keyword query within region R_0 with the radius r_0 and centered at $Q.p$, and obtain a set of check-in sub-sequences $SSet$ that contain query keywords by traversing the GIF component of HR index. The sub-sequences with the same user id in $SSet$ are merged to obtain the candidate sub-sequence CSS . The fully keyword-matching CSS push into PH_2 with l_{q_2} . For partially matching CSS , we expand the query range by increasing r_0 and utilize the same method to find sub-sequences (SS) in the region formed by range R_i but not include in the region R_{i-1} . When l_{q_2} of current CSS is smaller than the k -th entry in PH_2 , top- k mobile users are returned.

4.2 Candidate Verification

Since it is possible that the partly matching users is superior to k full matching users, we need to verify further the candidate set generated in the first phase to obtain the final results. We need complement the missing entry for the partly matching users. Obviously, the computational and I/O cost are high for this task. For the missing entry, we replace the truth value with the corresponding entry in GU . Given a partly matching u' and let $unsat_q$ be the unsatisfying query task, the $UppB(u', Q')$ as defined as follows.

$$UppB(u', Q') = \sum_{q_i \in Q' - unsat_q} l_{q_i} + \sum_{q_j \in unsat_q} GU.l_{q_j} \quad (7)$$

The candidate verification procedure is as follows. we firstly add the full matching users into the results and treat the k -th user's aggregate score as the pruning threshold τ . If the $UppB(\cdot)$ of partly matching users is smaller than the τ , these users are pruned without further computation. Otherwise, the algorithm has to retrieve the users of the corresponding query and compute the aggregate score. After processing all the partly matching users, the algorithm outputs the top- k users as the final result.

5 Experiments Evaluation

Our experiments are conducted on a real data set, Gowalla check-in dataset (GCI), as the users' check-in histories. GCI contains a total of 6,442,890 check-ins of 196,591 users and 950,327 edges of users' social graph over the period of Feb. 2009 - Oct. 2010. In our application scenario, we simulate a certain number of mobile users moving in the city center. At each update cycle, $r\%$ of the users in the memory are evicted by replacing with equal number of fresh users, their check-ins are generated randomly by GCI data set.

We implement all the algorithms in JAVA. All our experiments were executed on a windows platform with an Intel(R) Core(TM)2 Duo CPU of T7500 @ 2.66GHZ and 4GB RAM. The responding time and accuracy ratio are used as our performance metrics, where the accuracy ratio is defined as $Aratio = |Result_{CanGV} \cap Result_{CUFinder}| / k$.

We compare ***CanGV*** against ***CUFinder*** and ***baseline algorithm***. The baseline algorithm exploits the IR-tree to index the users' check-in histories. The IR-tree [4] integrates R-tree and the inverted file. The MBR and text of a check-in history *CH* is obtained by aggregating the MBR and the text information associated with each location in *CH*. Then, we utilize the LkT query algorithm [4] to find the top-k mobile users.

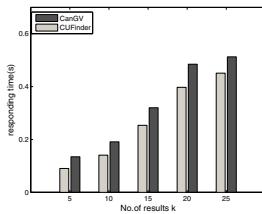


Fig. 2. Effect Of k (GCI)

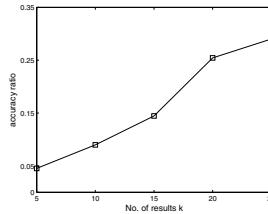


Fig. 3. Acc of CUFinder vs CanGV

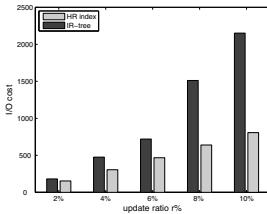


Fig. 4. Effect of $r\%$

In the first set of experiments, we study the responding time of *CanGV* and *CUFinder* and the accuracy of *CUFinder* by varying k . As shown in Fig.2 , the responding time of two methods increase as k grows. When k is small, two methods show the similar rate of increase. However, as k increases, the performance of *CanGV* diminishes since *CanGV* needs to access more users' check-in histories to obtain the *top-k* results. Fig.3 depicts the accuracy ratio of *CUFinder* relative to *CanGV*. We observe that the accuracy ratio slightly increases as k grows. This is because two methods will retrieve more same mobile users. Moreover, the average accuracy ratio can not reach 0.3. Therefore, our proposed query method is the best choice.

In the subsequent experiment, we evaluate the update performance of HR index and IR-tree for check-in histories, while they use the CU index for the current users. Fig.4 shows that the update performance of HR index notably outperforms IR-tree for all values of $r\%$. Due to the fact that HR index employs an adaptive multi-level grid that has an lower maintenance overhead.

6 Conclusion

In this paper, we study the problem of top- k potential trust friends (*PTF*) query problem , which aims to find top- k mobile users with the the highest demand satisfaction and trust score in the vicinity. We introduce a trust scoring model to measure users' trust degree. Furthermore, we present an efficient query processing algorithm based on *CUHR* index structure and exploit candidate generation-and-verification method to answer *PTF* query. Our experimental evaluation shows the efficiency of our algorithms.

References

1. Amir, A., Efrat, A., Myllymaki, J., Palaniappan, L.: Buddy track - Efficient proximity detection among mobile friends. In: INFOCOM, pp. 489–511 (2004)
2. Yiu, M., Hou, L., Saltenis, S., Tzoumas, K.: Efficient proximity detection among mobile users via self-tuning policies. In: VLDB, pp. 985–996 (2010)
3. Sidlauskas, D., Saltenis, S., Jensen, C.S.: Parallel main-memory indexing for moving-object query and update workloads. In: SIGMOD, pp. 37–48 (2012)
4. Cong, G., Jensen, C.S., Wu, D.: Efficient retrieval of the top- k most relevant spatial web objects. VLDBJ. 2(1), 337–348 (2009)
5. Han, J., Kamber, M.: Data mining: Concept and techniques, 2nd edn. Morgan Kaufmann (2006)

Sensitive Edges Protection in Social Networks

Liangwen Yu^{1,2,3}, Tao Yang^{1,2,3}, Zhengang Wu^{1,2,3},
Jiawei Zhu^{1,2,3}, Jianbin Hu^{1,2,3,*}, and Zhong Chen^{1,2,3}

¹ Institute of Software, School of EECS, Peking University, China

² MoE Key Lab of High Confidence Software Technologies (PKU)

³ MoE Key Lab of Network and Software Security Assurance (PKU)

{yulw,ytao,wuzg,zhujw,hjbin,chen}@infosec.pku.edu.cn

Abstract. With the increasing popularity of online social networks, such as twitter and weibo, privacy preserving publishing of social network data has raised serious concerns. In this paper, we focus on the problem of preserving the sensitive edges in social network data. We call a graph is k -sensitive anonymous if the probability of an attacker can re-identify a sensitive node or a sensitive edge is at most $\frac{1}{k}$. To achieve this objective, we devise two efficient heuristic algorithms to respectively group sensitive nodes and create non-sensitive edges. Finally, we verify the effectiveness of the algorithm through experiments.

Keywords: social network, privacy preserving, data publishing, sensitive edges.

1 Introduction

A social network can be modeled as a graph in which each node represents an individual, and the connections between individuals are summarized by the edges. An original graph G is shown in figure 1(a), and figure 1(b) demonstrates the naive anonymization G^* [1] which replaces all identifiers of individuals with randomized integers. However, this simple strategy can't resist attacks by the structural knowledge of individuals such as degree, neighborhood graph, and so on. Most of the previous works have been studied on protecting the node re-identification and edge re-identification in social networks. Recently, online social network sites of twitter and weibo have developed the new application of close friends, which means a person's close friends are confidential for participants in the social networks. For example, in figure 1(c), Bob and Carol are close friends, while Alice and Carol are regular friends. The sensitive edge represented by the dotted line is the sensitive information which needs to be protected in our paper.

Zheleva et al.[2] first considered a link re-identification attacks in which the adversary infers sensitive relationships from non-sensitive ones in the graph that contains multiple types of edges. Comparing with anonymous strategy in [2], our method has the following differences. First, the graph in [2] contains multiple types of edges which could simultaneously exist between two nodes. In our paper,

* Corresponding author.

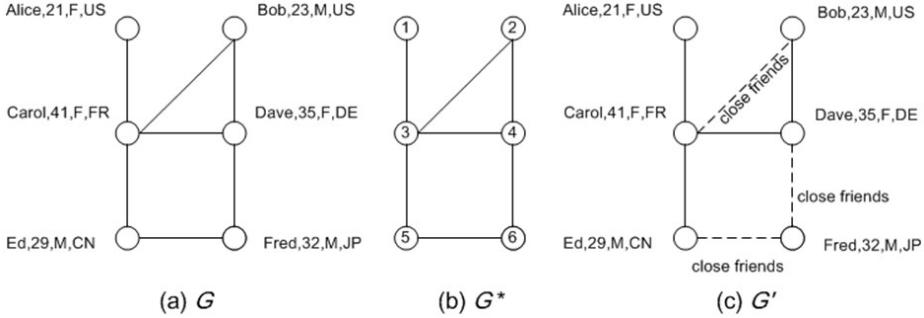


Fig. 1. A social network G , the naive anonymization G^* , a social network G'

there are two types(sensitive or non-sensitive) edges in the graph, and only one edge could exist between two nodes. Second, the node in our graph is described by a label, which is neglected in [2]. This label, generalized in the anonymization, is available for the data utility of aggregate network queries. Third, [2] applies clustering-based method in producing super nodes and super edges to protect relationships disclosure, while our method create some noise edges. Some graph properties, such as clustering co-efficient and average path length, couldn't be calculated in their anonymous results.

The remainder of the paper is organized as follows. Section 2 gives the problem definition. Section 3 introduces two anonymous algorithms. Section 4 reports experimental results. Section 5 concludes the paper.

2 Problem Definition

In this paper, a social network is modeled as an undirected simple graph $G(V, E^s, E^n, L, \mathcal{L})$, where V is a set of nodes, E^s is a set of sensitive edges, E^n is a set of non-sensitive edges, L is a set of labels and a function $\mathcal{L} : V \rightarrow L$ assigns each node a label. A label has an identity(such as user id or name), and a set of properties(such as age, gender and country). Let G^* denotes the anonymous graph of G .

The first step to anonymization is to know what external information of a graph may be acquired by an adversary. In a social network, an attack can easily view a person's partial profile and his connection information. So we assume the attacker's background knowledge as follows: An attacker knows a user's label and degree. For example, an attacker knows Bob is 23 years old with degree 2.

Definition 1 (sensitive node). *A node is called a sensitive node, if there are at least one sensitive edge connecting to it.*

We use V^s to denote the set of sensitive nodes. As each sensitive edge connects to two sensitive nodes in the graph, the sensitive nodes disclosure will increase the probability of the sensitive edges disclosure. For example, in figure 1(c), if an attack can re-identify Ed and Fred in the graph, he can also ascertain the

sensitive edge between them. To better protect the sensitive edges, our objective is to protect both sensitive nodes and sensitive edges in the anonymous graph.

Definition 2 (*k*-sensitive anonymity). A graph G^* is a k -sensitive anonymity if, for each sensitive node u in G^* , the probability that an attack re-identifies u is at most $\frac{1}{k}$; for any two nodes u_x and u_y , the probability that an attack can re-identify that there is a sensitive edge between them is at most $\frac{1}{k}$.

Generalization is to group nodes and enable the nodes within each group possess identical label. We use a similar method in [3] to calculate the generalization cost, and also use this cost to partition the sensitive nodes.

3 K-Sensitive Graph Anonymization

To protect edges, [4] focuses on masking the mapping via grouping the nodes of the graph. This technique retains the entire graph structure but perturbs the mapping from labels to nodes. They propose a *Safety Condition* that each node must interact with at most one node in any group and no edges exist in a group. [5] defines the *Edge Identification* which measures the likelihood of identifying an interaction. Both *Safety Condition* and *Edge Identification* ensure sparsity of interactions between nodes of any two groups, but *Safety Condition* is more restrictive than *Edge Identification*. Based on them, we limit the number of sensitive edges between any two groups while partitioning the sensitive nodes.

3.1 Sensitive Nodes Partition

We aim to protect both sensitive nodes and sensitive edges. Sensitive edges don't exist among non-sensitive nodes. So grouping is processed among sensitive nodes.

Definition 3 (Sensitive Safety Condition). A grouping of sensitive nodes in graph G , satisfies the Sensitive Safety Condition if

$$\forall(v, w) \in E^s : v \in g \wedge w \in g \Rightarrow v = w;$$

$$\forall \text{group } g_x, g_y, x \text{ is the number of sensitive edges between } g_x \text{ and } g_y, x \leq \frac{|g_x| \cdot |g_y|}{k}.$$

Theorem 1. If sensitive nodes partition satisfies Sensitive Safety Condition, the anonymous graph can protect sensitive edges against attacks with background knowledge of node's label.

Proof. The first condition in Sensitive Safety Condition constrains sensitive edges not exist in one group, the second condition constrains the number of sensitive edges x between any two groups is no more than $\frac{|g_x| \cdot |g_y|}{k}$. *Edge Identification* is the probability an attacker can attach to a particular pair of users between any two groups. With respect to sensitive nodes partition, sensitive *Edge Identification* equals to $\frac{|(g_x \times g_y) \cap E^s|}{|g_x| \cdot |g_y|} = \frac{x}{|g_x| \cdot |g_y|} \leq \frac{\frac{|g_x| \cdot |g_y|}{k}}{|g_x| \cdot |g_y|} = \frac{1}{k}$. So after generalization, the probability of attack with label's background knowledge can re-identify a sensitive edge is no more than $\frac{1}{k}$, which satisfies the sensitive edges privacy protection.

The algorithm SNP performs sensitive nodes partition. The input of SNP is an original graph G and a parameter k , and the output is a set of groups with size at least k . The groups are created one at a time. To form a new group, a sensitive node in V^s with maximum degree and not yet allocated to any group is selected as a seed for the new group(lines 3-4). Then the algorithm adds nodes into the group with the minimum generalization cost under *Sensitive Safety Condition* until the group size reaches k (lines 5-13). At each step, we calculate the group's candidate set under *Sensitive Safety Condition*, and add one sensitive node into the group. The selected one node has to be unallocated yet to any group and has the minimum generalization cost in candidate set. In some cases, the group size is less than k (lines 14-17). For each node in the group, we add it into the previous constructed group which has the minimum generalization cost for the node addition. The SNP ceases when there is no sensitive node left in V^s .

After grouping the sensitive nodes, we generalize the label of sensitive nodes in any group. In this way, using background knowledge of node's label, an attacker can't re-identify a sensitive edge with a probability higher than $\frac{1}{k}$.

Algorithm 1. Sensitive Nodes Partition (SNP) Algorithm

Input: An original graph $G(V, E^s, E^n, L, \mathcal{L})$ and the parameter k .

Output: A set of groups $S = \{g_1, g_2, \dots, g_m\}; \bigcup_{i=1}^m g_i = V^s; g_i \cap g_j = \emptyset, i, j = 1, 2, \dots, m, i \neq j; |g_i| \geq k, i = 1, 2, \dots, m.$

- 1: $S = \emptyset; i = 1;$
- 2: **repeat**
- 3: $SeedNode =$ the sensitive node with maximum degree from V^s ;
- 4: $g_i = \{SeedNode\}; V^s = V^s - \{SeedNode\};$
- 5: **repeat**
- 6: $CandidateSet = \emptyset;$
- 7: **for** each node $u \in V^s$ **do**
- 8: **if** inserting u into g_i doesn't violate sensitive safety condition **then**
- 9: $CandidateSet = CandidateSet \cup \{u\};$
- 10: **if** $|CandidateSet| > 0$ **then**
- 11: $u' =$ the sensitive node in $CandidateSet$ with minimum generalization cost of inserting it into g_i ;
- 12: $g_i = g_i \cup \{u'\}; V^s = V^s - \{u'\};$
- 13: **until** $((|g_i| = k) \text{ or } (|CandidateSet| = 0))$
- 14: **if** $|g_i| < k$ **then**
- 15: **for** each node $u \in g_i$ **do**
- 16: $g' =$ the group in S with minimum generalization cost of inserting u into it under sensitive safety condition;
- 17: $g' = g' \cup \{u\};$
- 18: **else**
- 19: $S = S \cup \{g_i\}; i++;$
- 20: **until** $|V^s| = 0$

3.2 Non-sensitive Edges Addition

To resist attacks with node's degree, we should enable all the nodes in the same group have the same degree. As sensitive edges are privacy information, we only create non-sensitive edges. First, we set a target degree for each group and put nodes whose degree needs to be increased into a *NodeList*(lines 1-7). In lines 8-19, we create non-sensitive edges between nodes in *NodeList*. Last, we randomly create non-sensitive edges between sensitive nodes and non-sensitive nodes.

Algorithm 2. Non-sensitive Edges Addition (NEA) Algorithm

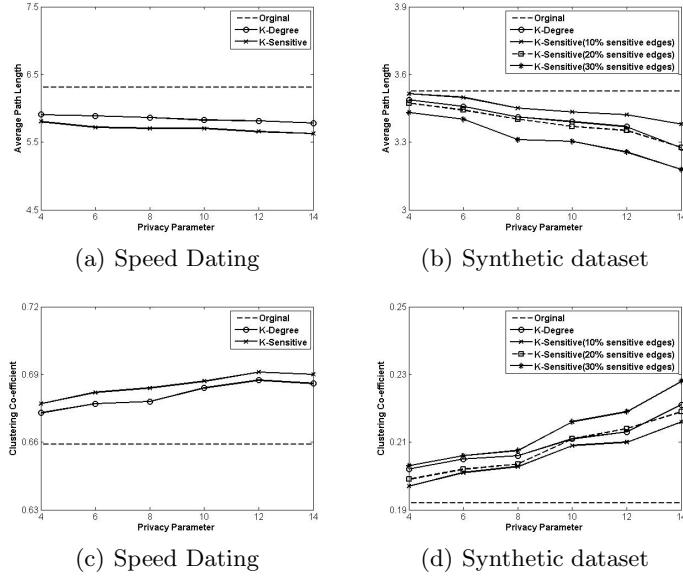
Input: A set of groups S . An interim graph G which is transformed from original one.
Output: An anonymous graph G^* .

```

1: NodeList =  $\emptyset$ ;
2: for each group  $g \in S$  do
3:   Target = the highest node's degree in  $g$ ;
4:   for each node  $u \in g$  do
5:     if  $u.degree < target$  then
6:        $u.number = Target - u.degree$ ;
7:       NodeList.add( $u$ );
8:   sort NodeList in the number descending order;
9: while NodeList.size() > 0 do
10:    $v = NodeList[0]$  and remove it from NodeList;
11:    $i = 0$ ;
12:   while  $((v.number > 0) \text{ and } (i < NodeList.size()))$  do
13:      $w = NodeList[i]$ ;
14:     if  $((v, w) \notin E^s) \text{ and } ((v, w) \notin E^n)$  then
15:        $E^n = E^n \cup (v, w)$ ;
16:        $v.number --$ ;  $w.number --$ ;  $i ++$ ;
17:     else
18:        $i ++$ ;
19:   update NodeList; //update the tuples in NodeList and sort NodeList
20:   if  $v.number > 0$  then
21:     randomly pick  $v.number$  non-sensitive nodes which don't connect to  $v$ , and
       create  $v.number$  non-sensitive edges between  $v$  and non-sensitive nodes;
```

4 Experimental Results

In this section, we study the utility of anonymous graphs from the average path length and clustering co-efficient on Speed Dating and a synthetic dataset. As the anonymous method in [2] alters the network structure, we couldn't compare our anonymous results with it. In this experiment, we just compare our approach with k -degree anonymity[6]. Figure 2 shows average path length and clustering co-efficient of anonymous graphs for $k = 4, 6, 8, 10, 12, 14$. Generally, all these observations verify that the k -sensitive anonymity could acceptably capture main features of the social network.

**Fig. 2.** Experimental results

5 Conclusions

In this paper, we protect sensitive edges in social network. We present two heuristic algorithms. Our experiments show anonymous graph has acceptable utility.

References

1. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In: Proceedings of the 16th International Conference on World Wide Web, pp. 181–190. ACM (2007)
2. Zheleva, E., Getoor, L.: Preserving the privacy of sensitive relationships in graph data. In: Bonchi, F., Malin, B., Saygin, Y. (eds.) PInKDD 2007. LNCS, vol. 4890, pp. 153–171. Springer, Heidelberg (2008)
3. Campan, A., Truta, T.M.: A clustering approach for data and structural anonymity in social networks. In: Privacy, Security, and Trust in KDD Workshop, PinKDD (2008)
4. Bhagat, S., Cormode, G., Krishnamurthy, B., Srivastava, D.: Class-based graph anonymization for social network data. Proceedings of the VLDB Endowment 2(1), 766–777 (2009)
5. Bhagat, S., Cormode, G., Srivastava, D., Krishnamurthy, B.: Prediction promotes privacy in dynamic social networks. In: Proceedings of the 3rd Conference on Online Social Networks, p. 6. USENIX Association (2010)
6. Liu, K., Terzi, E.: Towards identity anonymization on graphs. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 93–106. ACM (2008)

Ontology-Based Semantic Search for Large-Scale RDF Data

Xiaolong Tang^{1,2}, Xin Wang^{1,2,*}, Zhiyong Feng^{1,2}, and Longxiang Jiang^{1,2}

¹ School of Computer Science and Technology, Tianjin University, Tianjin, China

² Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China

{txl1290, lxjiang2012}@gmail.com, {wangx, zyfeng}@tju.edu.cn

Abstract. In recent years, the Web of Data has emerged with the release of growing amount of Linked Data. Since traditional Information Retrieval (IR) technologies are no longer suit for the retrieval on Linked Data, it becomes difficult for ordinary users to retrieve the data efficiently and accurately. This paper presents a method of doing keyword search on Web of Data. We propose two distributed inverted index schemes, one of which is built from Linked Data and the other from the ontology. And as a necessary part of the ontology index, an ontology encoding scheme is also proposed. Based on the index schemes, we design an improved ranking algorithm named OntRank by introducing a semantic factor into the BM25F ranking model. The experimental evaluation illustrates the efficiency of constructing indexes and the precision of retrieval results.

Keywords: Linked Data, keyword search, distributed inverted index.

1 Introduction

The Resource Description Framework (RDF)[1] is a standard data model for describing the Semantic Web whose goal is making the information machine-readable. Linked Data[3] refers to graph-structured data that encoded in RDF and accessible via Hypertext Transfer Protocol (HTTP). In recent years, with the development of Linked Data, the researchers focus on constructing semantic search engines to access more accurate knowledge by taking advantage of the graph structure of Linked Data. However, the realization of the semantic search engine implies two major challenges: the system must scale to large amount of data and the search on Linked Data must be efficient and accurate.

A number of semantic search engines have been developed in past few years. But our survey on these semantic search engines reveals that although they improve the traditional IR technologies, a lot of vulnerabilities lie in these systems. For example, some of them[2,4,5] are not suitable for naive users who are not necessarily proficient in the semantic data and the structured query language. The others[9,13,14,8,10] have considered the importance of user-friendly, but the accuracy of their query results is at a low-level.

* Corresponding author.

In order to address aforementioned issues, we present a distributed semantic keyword search approach, which can provide an efficient and accurate search on Linked Data. Our work can be regarded as an extension to the traditional IR probabilistic retrieval model. More specifically, the main contributions of this paper include:

- A distributed inverted index scheme is proposed for Linked Data. And based on an ontology encoding scheme that we present in this paper, we design a specialized inverted index scheme for the ontology.
- We devise a new ranking algorithm, called OntRank, which is designed to improve the traditional IR ranking algorithm by introducing a semantic factor into the ranking function.
- We perform comprehensive experiments to demonstrate the performances of our index schemes and the accuracy of the OntRank algorithm.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 describes the distributed inverted index schemes which are built from the RDF data and the ontology. The OntRank algorithm, which contains the semantic factor, will be proposed in Section 4. Section 5 reports the experimental results. In Section 6, we conclude and outlook the future work.

2 Related Work

Since Semantic Web develops rapidly in recent years, users look forward to making full use of the large amount of RDF data. For this purpose, lots of semantic search engines have been developed. Most of them are constructed based on the RDF query language, such as[2,4]. The characteristic of these search engines is that they provide a very complex structured query language, such as SPARQL[6] and SeRQL[11], to support RDF data accessing. However, they do not support keyword search on RDF data, so in order to access RDF data with these search engines, end users have to be quite familiar with the structure of RDF data and the syntax of structured query language.

To satisfy the requirements of ordinary end users, some keyword-based semantic search engines have been built. Currently, there are two main approaches on doing keyword search on RDF data: one of which uses the traditional IR technologies, the other one converts the keyword query into the structured query language. In the first approach, Semplore[8], K-Search[9] and Swoogle[10] combine the keyword retrieval with graph-structured RDF data, then compute a ranked list for the semantic documents. SWSE[13] is also a semantic keyword search engine. The main difference between SWSE and the aforementioned search engines is that, it provides an entity-centric search on RDF data. But all of them only adopt the traditional IR technologies to build the inverted index from RDF data, and they do not consider the semantic that exists in the keyword queries. SemSearch[14] is an example of the second approach, it transforms a keyword search into a structured SeRQL[11] query based on finding the matches of the two parts of the keyword search. However, this method may lose a lot of necessary matches when the length of the keywords is more than two.

Our work is developed based on Jingwei system[15] and it is quite different from the aforementioned search engines. First, our inverted indexes are built both from RDF data and the ontology. Second, our method can recognize the semantics in the queries. In our system, the keyword query contains two parts, the main keywords and the auxiliary keywords, such as *red apple@banana*. We get the semantics by checking the relations between the two kinds of keywords. Third, our approach can get more accurate results for the keyword queries according to the semantics.

3 Distributed Index Scheme

In this section, we will introduce two distributed inverted index schemes in our system, one of which is built from Linked Data, the other one is from the ontology. And we refer to the inverted index constructed on the assertional part of Linked Data as *A-index*, and the inverted index on the terminological part as *T-index*.

3.1 Distributed A-Index

Unlike traditional unstructured data, RDF data has no clear boundaries on documents. However, documents are the essential elements for inverted indexes, so we first give the definition of *RDF documents*.

Definition 1. Assume that there are two disjoint infinite sets U and L , where U is the set of RDF URIs and L is the set of RDF literals. An RDF triple is a tuple of the form

$$(s, p, o) \in U \times U \times U \cup L$$

In this tuple, s is called the subject, p the predicate, and o the object.

Definition 2. An RDF graph is a finite set of RDF triples. Let G be an RDF graph. 1) A subgraph of G is a subset of G . 2) $\text{subj}(G)$ denotes the set of all subjects in G .

Definition 3. Given an RDF graph G and a subject $s \in \text{subj}(G)$, an RDF document for s is a subgraph D_s of G where $\text{subj}(D_s) = \{s\}$.

In Definition 3, we define the *RDF documents* as the collection of triples with the same *subject*, and the documents ID is s . From the traditional IR viewpoint, the documents have several fields, such as title, abstract and text, and each of them has different weights. Similarly, *RDF documents* also have the concept of fields, which is divided based on the meaning of p . Since BM25F[16], the variant of BM25, has considered the influence of fields in the ranking algorithm, we decide to devise our *A-index* on top of BM25F. To satisfy the BM25F ranking algorithm, we have divided RDF data into 4 fields, i.e. *label*, *comment*, *type* and *others*, and the information of the fields will be stored in *A-index*. Figure 1 shows

Algorithm 1. Construction of *A-index*

```

Input: RDF Data
Output: A-index
1: map((s,p,o)):
2: f = {label, comment, type, others}
3: if o ∈ L then
4:   if filed(p) == f then
5:     for each term ∈ o do
6:       EMIT((term,s),(f,1))
7:     end for
8:   end if
9: end if
10: reduce((term,s),iterator values):
11:   for value in values do
12:     field = value.getLeft()
13:     frequency = field.getSum()
14:     hashtable.put(field,frequency)
15:   end for
16: set A-index { RK ← term, CN ← s, CV ← hashtable }

```

the structure of *A-index*, which is a 3-layer key-value structure, and the name of them are the row key (RK), the column name (CN), the column value (CV). In the design of *A-index*, we consider both efficiency and scalability, thus the MapReduce framework is adopted to build the index on top of Cassandra that is a distributed key-value NoSQL database. The constructing process of *A-index* is shown in Algorithm 1.

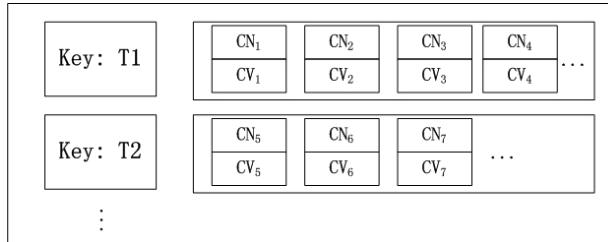


Fig. 1. The structure of A-index

In the map phase, we only process the triple (s, p, o) whose o is in L . In lines 3-9, we get the field ID and the document ID, for each term, and then emit them to the reduce function. And in the reduce phase, we first receive the term and document ID from map. Then in lines 11-15, we use a hash table to collect the detailed term frequency information, the key of the hash table is the field ID and the value is the corresponding term frequency. Line 16 inserts the term, the document ID and the hash table into the *A-index*.

3.2 Distributed T-Index

Although the *A-index* can accomplish the mission of doing keyword search on RDF data, the semantics in RDF data is not exploited. Thus, we build the *T-index* as a supplement of the *A-index* by indexing the ontology. And in order to use the ontology conveniently, an ontology encoding scheme is also proposed. This section will introduce the expression of the ontology and the construction of the *T-index* in detail.

3.2.1 Ontology Encoding

As we know, the ontology contains abundant semantics, which can be applied to express relations of entities. However, recognizing relations of entities is quite difficult if we directly store and use the ontology. In addition, the ontology is usually updated to enrich its semantics, so the management of the ontology will have a high cost. ORDPATH[7] is a hierarchical coding scheme whose benefits are two-fold. The first one is that the ORDPATH scheme allows inserting new nodes at arbitrary positions in the tree structure without relabeling any old nodes, so the update costs of ORDPATH will be at a low-level. The other one is that the code of ORDPATH can be compressed into a binary form which is easy for comparing ORDPATH values. In this way, we can recognize whether two nodes are child-parent nodes or sibling nodes by only comparing their binary code length. Obviously, ORDPATH is appropriate to ontology encoding, and the application of ORDPATH can represent the subsumption relationship appropriately. Thus, we adopt ORDPATH as the ontology encoding scheme.

Example 1. In Fig. 2, G is a part of the tree structure of entity classes which encoded with ORDPATHs. We can see that there are five layers from the root node *owl:Thing* to the leaf node *Pine* and the corresponding ORDPATHs code is under them. It is easy to identify the subsumption relationship of entity classes based on the comparison of ORDPATH. The code of *Plant* is 1.25.3, and we can infer that its ancient nodes is encoded with 1, 1.25, its sibling node is 1.25.5

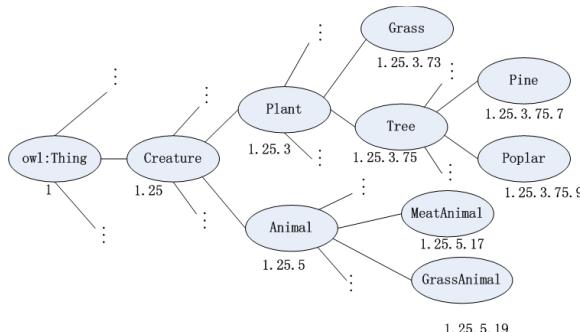


Fig. 2. The inheritance relationship of entity classes encoded by ORDPATHs

and one of its children nodes is 1.25.3.75, the corresponding entity classes are *owl:Thing*, *Creature*, *Animal* and *Tree*. According to the insertion strategy of ORDPATH, we can insert new nodes 1.25.1 to the left of *Plant* and 1.25.7 to the right of *Animal*. In addition, we can also insert the nodes at arbitrary positions, such as the position between *Plant* and *Animal*, with an even integer. 1.25.4.1 is an example of sibling nodes between *Plant* and *Animal*, and we can see that the even integers do not count for ancestry: 1.25.4.1 is a child of 1.25. This property shows that ORDPATHs is scalable and the old nodes need not to be relabeled when we update ORDPATHs, so it has a high update performance.

3.2.2 T-index Architecture

Definition 4. Given a set of all terms T , an RDF document D and an entity class C . t is an element of T , if the entity class of $\text{subj}(D)$ is C and t is in D , then we call that t relates to the entity class C , which is expressed as $t \sim C$, and we use $\text{class}(t)$ to express the set of entity classes that relate to t . For T_i that is a subset of T , if $t \sim C$ and $t \in T_i$, we call $T_i \sim C$. We define $\text{CLASS}(T_i) = \bigcup_{t_i \in T_i} \text{class}(t_i)$.

Since each s has at least one entity class and the relation between the entity classes can describe the relation between the corresponding subjects, we consider that a similar correlation exists between the terms of RDF documents and entity classes. Thus, we decide to construct the *T-index* according to the relation between terms and entity classes, which is defined in Definition 4. The index structure and the method for creating the *T-index* are like what we discussed in section 3.1. But a little difference between them is that the construction of the *T-index* has two mappers and two reducers. Algorithm 2 is the pseudo-code of building the *T-index*. The inputs are two files in DBpedia, where the o in the *instance_type* is the type of the s and the o in the *short_abstract* is the abstract of the s .

In the map phase of the first MapReduce, lines 2-7 identify which file the triple (s, p, o) belongs to, and then emit the s , o and an identifier to the reducer. Then in the corresponding reduce phase, lines 10-13 get the most accurate ORDPATH of the entity classes of the s , when o is the type of s . After that, we emit the ORDPATHs and the abstracts of the same s as an intermediate file, which is the input of the second MapReduce job. And in the second MapReduce job, we first emit each term in the abstracts and the corresponding ORDPATHs in lines 20-22. Then lines 24-26 make a statistic on $\text{class}(t)$ for each term t , which contains the ORDPATHs and the corresponding weights. Finally, we insert the terms, the ORDPATHs and the weights into the *T-index*.

Algorithm 2. Construction of T -index

Input: *instance_type, short_abstract*

Output: *T-index*

```

1: first_map((s,p,o)):
2: if (s,p,o) $\in$  instance_type then
3:   EMIT(s,(o,type))
4: end if
5: if (s,p,o) $\in$  short_abstract then
6:   EMIT(s,(o,term))
7: end if
8: first_reduce(s,iterator values):
9: for value in values do
10:   if value.getRight() == type then
11:     subj_type = value.getLeft()
12:     ordpaths = subj_type.getOrdpaths()
13:   end if
14:   if value.getRight() == term then
15:     text == value.getLeft()
16:   end if
17: end for
18: EMIT(ordpaths, text)
19: second_map(ordpaths, text):
20: for each term  $\in$  text do
21:   EMIT(term,(ordpaths,1))
22: end for
23: second_reduce(term,iterator values):
24: for value in values do
25:   ordpaths = value.getLeft()
26:   calculate weight w of ordpaths
27:   set T-index { RK  $\leftarrow$  term, CN  $\leftarrow$  ordpaths, CV  $\leftarrow$  w }
28: end for

```

4 Semantic Ranking Algorithm

Based on the indexes built in section 3, we propose an improved ranking algorithm, called OntRank, which take advantage of the relation between the data to improve the effectiveness of document ranking.

4.1 RO Calculation

Definition 5. Given a set of entity classes C , the entity classes C_0, C_1 and $C_2 \in C$, and C_0 is the minimum common parent class of C_1 and C_2 . If the main keywords $T_1 \sim C_1$ and the auxiliary keywords $T_2 \sim C_2$, the distance between C_1 and C_2 through C_0 is called RO (Rank Order), which reveal the relevance of T_1 and T_2 , and expressed as $RO(C_1, C_2)$. $RO(T_1, T_2)$ is a set of $RO(C_i, C_j)$ that $C_i \in CLASS(T_1)$ and $C_j \in CLASS(T_2)$.

Algorithm 3. Calculating $RO(T_1, T_2)$

Input: the main keywords T_1 , the auxiliary keywords T_2
Output: rank order $RO(T_1, T_2)$

```

1: setofOrdpAthA =  $T_1.getOrdpAths()$ ;
2: setofOrdpAthB =  $T_2.getTopKOrdpAths()$ ;
3: for each ordpathsA in setofOrdpAthA do
4:   for each ordpathsB in setofOrdpAthB do
5:     ordpathsC = getMiniCommonOrdpAths(ordpathsA, ordpathsB);
6:      $RO(A, B) = |ordpaths\ A - ordpaths\ C| + |ordpaths\ B - ordpaths\ C|$ ;
7:     if  $RO(A, *)$  not in  $RO(T_1, T_2)$  then
8:       add  $RO(A, B)$  to  $RO(T_1, T_2)$ 
9:     end if
10:    if  $RO(A, *)$  in  $RO(T_1, T_2)$ , but  $RO(A, B) < RO(A, *)$  then
11:      modify  $RO(A, *)$  to  $RO(A, B)$ 
12:    end if
13:  end for
14: end for
15: return  $RO(T_1, T_2)$ 

```

In Definition 5, the main keywords and the auxiliary keywords are the two parts of our keyword query introduced in section 2, and both of them are sets of terms, so they have the characteristics of Definition 4. Since the semantics in the queries is essential to our ranking algorithm, we define RO as the semantic factor which is calculated by the correlation of entity classes related to the keywords. The corresponding pseudo-code of calculating $RO(T_1, T_2)$ is shown in Algorithm 3.

In Algorithm 3, we obtain the corresponding ORDPATHs of $CLASS(T_1)$ in line 1. To avoid too many classes related to the auxiliary keywords that may influence the query accuracy, line 2 gets the top- k correlative ORDPATHs based on the weights of ORDPATHs. Lines 3-14 calculate RO . Line 5 gets the minimum parent class of the chosen classes, such as A and B . Then in line 6, we calculate the minimum path length between A and B . In case 1, the hash table $RO(T_1, T_2)$ do not have the element of $RO(A, *)$, where $*$ can be any ORDPATHs of $setofOrdpAthB$, then we add $RO(A, B)$ into the hash table as $RO(A, *)$. In lines 10-12, although $RO(T_1, T_2)$ have the element of $RO(A, *)$, the minimum path length between A and B is less than $RO(A, *)$, we update $RO(A, *)$ with $RO(A, B)$.

Obviously, the time complexity of Algorithm 3 is $O(|M| \times |N|)$, that $|M|$ is the number of $CLASS(T_1)$ and $|N|$ is the number of top- k $CLASS(T_2)$. The main space cost is the storage of entity classes which related to keywords. Hence, the overall space complexity is $O(|M| + |N|)$.

Example 2. G (Fig.2) in section 3.2.1 is an example of entity classes which encoded with ORDPATHs. We suppose that $CLASS(T_1)$ is $\{Poplar, Grass\}$ and $CLASS(T_2)$ is $\{Pine\}$. First, we get the shortest path between $Poplar$ and $Pine$, which is $Poplar$ - $Tree$ - $Pine$, so $RO(Poplar, Pine) = 2$, then we add it to $RO(T_1, T_2)$. Similarly, we can also conclude $RO(Grass, Pine) = 3$ when $T_1 \sim Grass$ and $T_2 \sim Pine$. In this condition, $RO(T_1, T_2) = \{(1.25.3.75.9, 2), (1.25.3.73, 3)\}$.

4.2 OntRank Algorithm

Algorithm OntRank uses RO as the semantic factor to improve the effectiveness of the BM25F ranking function. In this section, a detailed description will be presented, which is about calculating the score of documents.

First of all, the BM25F ranking function should set several boost factors before calculating the score of documents. In our system, the boost factors are: $label = 30.0$, $comment = 5.0$, $type = 10.0$, $others = 1.0$. Moreover, BM25F also requires additional parameters, whose value is $K_1 = 4.9$, $b_{label} = 0.6$, $b_{comment} = 0.5$, $b_{type} = 0.5$ and $b_{others} = 0.4$. And in this paper, we use $score^{BM25F}(Q, D)$ to express the score of the query which based on the BM25F ranking algorithm. So the score of the OntRank ranking algorithm is calculated by

$$score^{OntRank}(Q, D) = score^{BM25F}(Q, D) \times (1 + \frac{1}{RO+1})$$

where RO is the concept defined in section 4.1.

In our method, RO is a list of non-negative integers, and with the value of RO decreasing, the correlation between the main keywords and the auxiliary keywords becomes higher. Here, we use $RO + 1$ to avoid the condition that $RO = 0$. Moreover, it can be seen that $\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots\}$ are the values of different $\frac{1}{RO+1}$, the differences between neighboring values are $\{\frac{1}{2}, \frac{1}{6}, \frac{1}{12}, \dots\}$. It is obvious that the value of RO improves more significantly when the relationship between the two kinds of keywords is getting closer. Meanwhile, the class of a returned document must be one of the classes related to the main keywords. It means that RO is also the relationship between returned documents and the auxiliary keywords. Thus, the score of documents can be distinguished clearly according to the different relevance levels of the documents and the auxiliary keywords.

5 Evaluation

In this section, we describe our evaluation method on the proposed indexes and algorithms, and show the experimental results. Since the solutions mentioned in the INEX 2009 is designed for semi-structured data and up to now we have not found a open source semantic search engine for RDF data, we only compare our method with the BM25F model.

5.1 Setting

All experiments are carried out by using a 4-nodes cluster as the underlying storage system. Each node has Intel Q8400 CPU and 8 GB memory. The operating system is Ubuntu 10.04. The database is Cassandra 0.8.1, and the Cassandra heap was set to 4GB. We use Apache Tomcat 7.0 as the Web server.

5.2 Dataset and Queries

For the evaluation we use the DBpedia 3.7 dataset excluding pagelinks; each characteristic of the DBpedia dataset is listed in Table 1. Although we only use

the DBpedia, our method is not optimized for the dataset. In other words, our method is general enough to be used for other datasets.

Table 1. Dataset characteristics

name	value
distinct triples	43,853,549
distinct subjects	8,870,118
distinct predicates	1,299
distinct objects	18,286,543

For the reasons that a) we adopt the INEX evaluation framework to provide a mean for evaluating the semantic keyword search approach b) Wikipedia is the source of the INEX collection. We have to preprocess the dataset that is build from the intersection of DBpedia and the INEX-Wikipedia collection[12]. The result is listed in Table 2.

Table 2. Dataset preprocessing result

dataset name	entity number
DBpedia	364,5384
Wikipedia	266,6190
Intersection(DBpedia, Wikipedia)	244,9229

The query set in our evaluation framework is taken from the INEX 2009 contest. INEX2009 offers the judgments of 68 queries. Each query consists of three parts with the same topic: title, description and narrative. In our experiments we adjust the titles by adding the semantics of descriptions and narratives into them. In this way the query will be more expressive and fit better to the semantic search. The average length of the queries is around 4 words. However, not all the semantics of queries suit for the OntRank algorithm, we take 50 of them as the final query set.

We take the IR metrics as the evaluation criterion. These metrics are generally oriented in two main directions: the ability to retrieve relevant documents and the ability to rank them properly. In our experiment, we use part of them: Mean Average Precision (MAP), which provides a single-figure measure of quality across recall levels; Geometric Mean Average Precision (GMAP) that is a variant of MAP by using a geometric mean; Precision after K documents (P@K) which measures the precision after K documents have been retrieved; and R-Precision that measures the precision after R documents have been retrieved, where R is the total number of relevant documents for a query.

5.3 Results: Indexing Cost

We performed an experiment on measuring the indexing cost with the increase of the number of nodes. The results are shown in Figure 3. Note that the indexing cost of *T-index* is the total executing time of two MapReduce jobs. From

the results, we observe that the time of constructing index will decrease by increasing the number of nodes. However, because of the information exchange between nodes, the downward trend of executing time are not linear. In addition, the scalability of our system is also reflected in the results, the problem of the explosive growth of Linked Data can be resolved merely by increasing the computer numbers.

5.4 Results: Query Accuracy

The query accuracy is measured by using a tool named trec_eval[17]. This tool is implemented for producing the IR metrics. Figure 4 shows the evaluation results obtained by two different methods, one is with OntRank and the other is only use the BM25F ranking function.

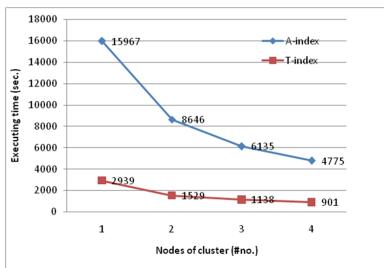


Fig. 3. System indexing cost

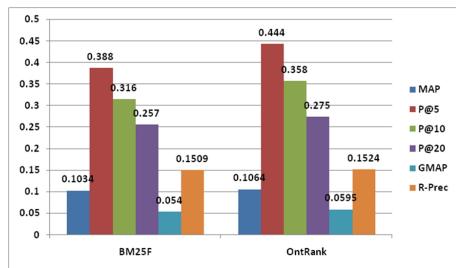


Fig. 4. IR metrics of BM25F and OntRank

The results draw from the experiments show that the OntRank ranking algorithm is better than BM25F in every single measure, especially P@K. This is not surprising since we add the semantics into the queries, so the search system can return more relevant documents after K documents have been retrieved. We can see that OntRank obtains the best performance, although some of the IR criteria do not improve significantly, such as MAP, GMAP and R-Prec. We have analyzed the reason for it, and find that there are many factors restricting the improvement of the results:

1. The relevant documents of some queries is about concepts, such as theories or algorithms, and their entity class is owl:Thing, so we can not get the corresponding auxiliary keywords, and the OntRank ranking algorithm remains ineffective.
2. Some entities do not have abstract, so some relevant documents may not be found, the OntRank ranking algorithm has no effect on them.
3. There are some mistakes in the classification of DBpedia, for example, the entity of Pavarotti belongs to dbpedia:Person in DBpedia, but its proper class is dbpedia:Musicalartist. And what is worse, this phenomenon is prevalent in the querying process. Hence, these mistakes will have a significant influence on the *RO* value which is the main impact factor of the IR metrics.

Fortunately, all of them are due to the quality of the dataset. So we can believe that with the improvement of the quality of the dataset, the IR evaluation criterion will improve more significantly.

6 Conclusion and Future Work

In this paper, we study the problem of doing keyword search on large-scale RDF data, and propose an effective approach to get more accurate query results. To support keyword search, we use MapReduce to build the inverted indexes both from Linked Data and the ontology. And in order to build the *T-index*, we design an ontology encoding scheme. Based on the indexes, we present a new ranking algorithm by adding the semantic factor, which is calculated by the relation between the main keywords and the auxiliary keywords, into the BM25F ranking function. And through the experimental results, we confirm the efficiency and the accuracy of our approach. In the future, we will keep on optimizing the OntRank algorithm. Moreover, we will seek a more granular knowledge base, which has a more accurate classification on entities, to compensate for the deficiencies of the existing DBpedia ontology data.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (Grant No. 61100049, 61070202) and the National High-tech R&D Program of China (863 Program) (Grant No. 2013AA013204).

References

1. Klyne, G., Carroll, J., McBride, B.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation. W3C (2004)
2. Corby, O., Dieng-Kuntz, R., Faron-Zucker, C.: Querying the Semantic Web with Coresse Search Engine. In: Proceedings of 15th ECAI/PAIS, Valencia, ES (2004)
3. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
4. Shadbolt, N., Gibbins, N., Glaser, H., et al.: Cs Aktive Space or how we learned to stop worrying and love the Semantic Web. IEEE Intelligent Systems, 41–47 (2004)
5. Zhang, L., Yu, Y., Zhou, J., Lin, C., Yang, Y.: An Enhanced Model for Searching in Semantic Portals. In: WWW (2005)
6. SPARQL 1.1, <http://www.w3.org/TR/2012/PR-sparql11-overview-20121108/>
7. O’Neil, P., O’Neil, E., Pal, S., Cseri, I., Schaller, G., Westbury, N.: ORDPATHs: Insert-Friendly XML Node Labels. In: Proceedings of ACM Conference on Management of Data (SIGMOD), pp. 903–908 (2004)
8. Wang, H., Liu, Q., Penin, T., Fu, L., Zhang, L., Tran, T., Yu, Y., Pan, Y.: Semplore: A Scalable IR Approach to Search the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 177–188 (2009)
9. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid Search: Effectively Combining Keywords and Semantic Searches. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 554–568. Springer, Heidelberg (2008)

10. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: A Search and Metadata Engine for the Semantic Web. In: Proc. of the 13th ACM CIKM Conf. ACM Press, New York (2004)
11. SeRQL, <http://www.w3.org/2001/sw/wiki/SeRQL>
12. Perez-Aguera, J.R., Arroyo, J., Greenberg, J., et al.: INEX+DBpedia: A Corpus for Semantic Search Evaluation. In: WWW, pp. 1161–1162 (2010)
13. Hogan, A., Harth, A., Umbrich, J., et al.: Searching and Browsing Linked Data with SWSE: The Semantic Web Search Engine. Web Semantics: Science, Services and Agents on the World Wide Web, 365–401 (2011)
14. Lei, Y., Uren, V., Motta, E.: SemSearch: A Search Engine for the Semantic Web. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 238–245. Springer, Heidelberg (2006)
15. Wang, X., Jiang, L.: Jingwei+ A Distributed Large-scale RDF Data Server. In: Proceedings of Asia-Pacific Web Conference (2011)
16. Stephen, R., Hugo, Z.: The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval, 333–389 (2009)
17. Trec_eval, <http://trec.nist.gov/treceval/>

A Unified Generative Model for Characterizing Microblogs' Topics

Kun Zhuang, Heyan Huang, Xin Xin, Xiaochi Wei, Xianxiang Yang,
Chong Feng, and Ying Fang

School of Computer Science and Technology
Beijing Institute of Technology, Beijing, China

{kzhuang, hhy63, xxin, wxchi, yangxianxiang, fengchong, yfang}@bit.edu.cn

Abstract. In this paper, we focus on the issue of characterizing microblogs' topics based on topic models. Different from dealing with traditional textual media (such as news documents), modeling microblogs has three challenges: 1) too much noise; 2) short text; and 3) content incompleteness. Previously, all these limitations have been investigated separately. Some work filters the noise through a prior classification; some enhances the text through the user's blog history; and some utilizes the social network. However, none of these work could solve all the above limitations simultaneously. To solve this problem, we make a combination of previous work in this paper, and propose a unified generative model for characterizing microblogs' topics. In the proposed unified approach, all the three limitations could be solved. A collapsed Gibbs-sampling optimization method is derived for estimating the parameters. Through both qualitative and quantitative analysis in Twitter, we demonstrate that our approach consistently outperforms previous methods at a significant scale.

Keywords: Latent Dirichlet Allocation, Microblog Analysis, Modeling Topics from Social Network Data.

1 Introduction

Microblog services, such as Twitter, Sina Weibo, and etc., have become one of the most popular social media. According to Infographics Labs¹, everyday in 2012 from Twitter², 1 million new users are registered, and 175 million tweets are generated on average. From the statistics from Marketing Land³, 32% of the Web users are using Twitter. The revenue obtained from the advertising service is predicted to be 540 million US dollars by 2014⁴. People use microblog services to broadcast information, have conversations, and provide comments [7,8,12]. Due to their impacts on the Web, many data mining services are conducted on this

¹ <http://www.mediabistro.com/alltwitter>

² <https://twitter.com>

³ <http://marketingland.com>

⁴ <http://www.webanalyticsworld.net>

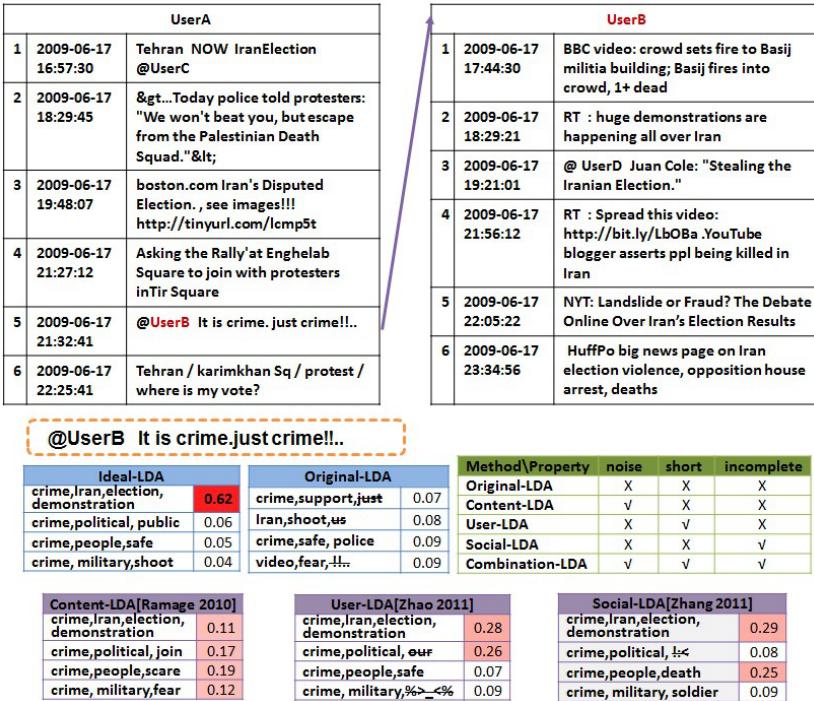


Fig. 1. A motivating example for characterizing a microblog's topic

new kind of social media, such as theme evolutionary analysis, public sentiment analysis, and etc. Topic analysis of microblogs is the fundamental problem for all these up-level services. Thus it has attracted much attention from both industry and academia [6,10,11,14,15].

In this paper, we focus on the issue of characterizing the microblogs' topics based on Latent Dirichlet Allocation (LDA) [1]. Fig. 1 illustrates an example of a microblog from Twitter. *UserA* posts six tweets. All the tweets are about events happened in the Iranian election in 2009. We utilize the 5th tweet, “@UserB It is crime. just crime!!!..”, as an example. This tweet talks about a violent demonstration event during the Iranian election period. An ideal characterizing model would project this microblog to the topic related to “crime, Iran, election, demonstration”, as shown in the Ideal-LDA table. In the table, four topics are shown. All of them are about the topic “crime”, which is the keyword occurred in the original microblog. But the other three (such as “general political crime”, “military crime”, and “public safety”) are not relevant to the demonstration event. Thus in an accurate topic analysis system, these three topics would not be assigned for the microblog. We utilize the red color to denote large assignment probability value; the pink color to denote the medium probability value; and the white color to denote the small probability value.

Different from traditional social textual media (such as news documents) which could be modeled well by the original LDA model [1], microblogs have three characteristics, which makes modeling their topics a challenging task. 1) **Too much noise.** The rate of noise words in microblogs is much higher than that in news documents. Such noise words include the stopword, the symbol, and etc. In our example, among the seven words (“*UserB*”, “it”, “is”, “crime”, “just”, “!”, “..”), only the word “crime” has a practical meaning for topic analysis. 2) **Short text.** As indicated from the name “microblog”, the length of the text is much shorter than that in news documents. From the statistics in Twitter⁵, most microblogs have less than 28 characters. 3) **Content incompleteness.** Since social network is playing an important role in microblogs, many texts are not self-contained. Instead, they rely on other texts in the social network by the relation of @users, re-tweets, and etc. Take the same example in Fig. 1, the tweet links *UserB*. From the tweets of *UserB* in the same time period, it could be inferred that the crime in the original tweet refers to the violent demonstration in the Iranian election. However, the original tweet itself does not contain the complete information. In summary, these three characteristics of microblogs make the original LDA model fail to work. We illustrate the result from the original LDA model in the Original-LDA table. It shows that the topics are not accurately modeled and assigned.

Although previous work overcomes some of these challenges separately, there are rarely any methods breaking all these limitations simultaneously. In the Content-LDA model [10,11], to solve the *noise* limitation, the non-content information is filtered by a prior classification process. Consequently, as shown in the Content-LDA table, the topics are modeled with much less noise. But since the text is too short and incomplete, the model cannot identify that the microblog is related to the demonstration event in Iranian election. Thus all the four topics related to “crime” would be assigned. In the User-LDA model [15], to solve the *short text* limitation, it utilizes the user’s whole microblog set to enhance the calculation accuracy of each microblog’s topic distribution. As shown in the User-LDA table, according to the user’s history, two non-relevant topics are filtered out, and only the other two, the demonstration in Iranian election and political events, are retained. However, it cannot further identify the demonstration event. In the Social-LDA model [14], to solve the *content incompleteness* limitation, when modeling the topic for a microblog, the texts in both the original microblog and the linked users are considered. As shown in the Social-LDA table, by referring the tweets from *UserB*, the demonstration event could be identified for the word “crime” in the original microblog. But as more information is integrated, some non-relevant information would be also added, which would confuse the model. Thus by considering the social network alone, the topic would also not be modeled accurately. In addition, noise exist in topics generated by the User-LDA model and Social-LDA model. We summarize the comparisons among different previous methods in the table in Fig. 1. It can be concluded that none

⁵ <http://www.smk.net.au/article/the-average-tweet-length-is-28-characters-long-and-other-interesting-facts>

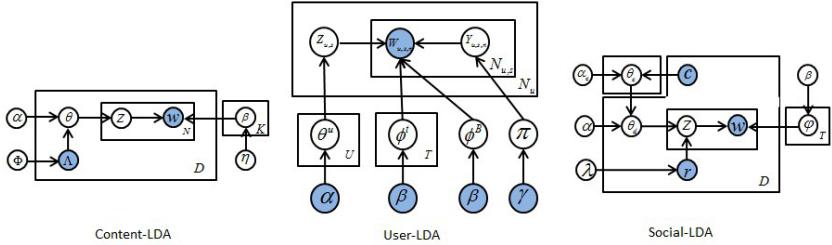


Fig. 2. The probabilistic graph of the three previous methods

of these methods overcomes all the limitations simultaneously. This motivates us to combine the ideas of all these methods into a unified topic model.

In this paper, we make combination of the ideas from the above three methods, and propose a unified generative model for characterizing microblogs' topics. This model could simultaneously solve the above three problems of microblogs, including noise, short text, and content incompleteness, in order to enhance the performance of modeling the topics. The main contribution lies in that we propose the generative process for the unified combination model, which is called “Combination-LDA”, and derive a collapsed Gibbs-sampling optimization method for estimating the parameters. Through experimental verification on the Twitter dataset, we demonstrate the effectiveness of our approach from both qualitative analysis and quantitative analysis. The proposed approach outperforms the above three methods significantly based on the metrics of log-likelihood and perplexity.

2 Background and Related Work

Topic models for traditional media have been well developed, from the Latent Semantic Analysis (LSA) [3], to the probabilistic Latent Semantic Indexing (PLSI) [5], and finally to the Latent Dirichlet Allocation(LDA)[1]. Topic modeling for microblogs, however, is a recent research task. Three representatives include the Content-LDA [10], the User-LDA [15], and the Social-LDA [14]. Other work include [2,6,8,9,12].

The generated process of the Content-LDA is shown in Fig. 2 (left). It extends the LDA by incorporating the supervision of a layer of labels, denoted by Λ , which is set beforehand. A prior classification is conducted in this model. It classifies the words into four categories, including the substance word, the status word, the style word, and the social word. In this way, the topic distribution is calculated with words of non-content categories filtered out.

The generated process of the User-LDA is shown in Fig. 2 (middle). It extends the LDA by incorporating the user's tweet history. Each user has a distribution over all the topics, which is denoted as θ_u . The distribution is generated from his whole micorblog set. Then for each microblog, the topic of each word is

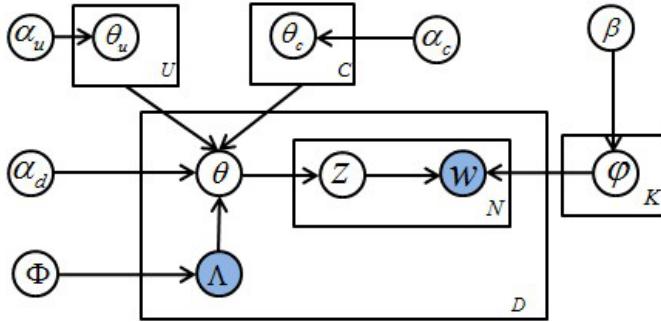


Fig. 3. The probabilistic graph for the Combination-LDA

assigned based on both the user's topic distribution and the microblog's topic distribution.

The generated process of the Social-LDA is shown in Fig. 2 (right). It extends LDA by utilizing the user's social relationships. Similar to the User-LDA model, each user has his own topic distribution. Each microblog has a set of linked neighbors from the social network, denoted by C . If the microblog contains @ relationships, its topic would be influenced by two parts, the content itself and the topic distribution from the social network.

3 The Proposed Unified Approach

The glossary of notations used for the model is shown in Table.1. Fig. 3. presents the probabilistic graphic model of the Combination-LDA.

Like the traditional LDA model, it is assumed that there is a word distribution φ_k corresponding to each k latent topic, and they follow the Dirichlet distribution with the prior β as

$$\varphi_k = (\varphi_{k,1}, \dots, \varphi_{k,n})^T \sim Dir(\cdot | \beta). \quad (1)$$

It can be observed from Fig. 3, the topic distribution of each microblog is influenced by four aspects. The part of the microblog content itself is essential. They follow the Dirichlet distribution with the hyper-parameter α_d as

$$\theta_d = (\theta_{d,1}, \dots, \theta_{d,K})^T \sim Dir(\cdot | \alpha_d). \quad (2)$$

The other three parts are breaking microblogs' three limitations as follows.

To solve the noise problem, we set labels manually. According to [10,11], we set three labels, the content word, the stopword, and the symbol. For each microblog d , the set of labels Λ_k^d is characterized by the hyperparameter Φ_k as

$$\Lambda_k^d \in \{0, 1\} \sim Bernoulli(\cdot | \Phi_k). \quad (3)$$

Table 1. Notation used in this paper

SYMBOL	DESCRIPTION
K	Number of topics
D	Number of microblogs
N	Number of words
U	Number of users
C	Number of social relationships
α_d	Hyperparameters for θ_d
α_u	Hyperparameters for θ_u
α_c	Hyperparameters for θ_c
θ_u	Topic distribution over user u
θ_c	Topic distribution over relationship c
θ_d	Topic distribution over the content of microblog d
$\theta^{(d)}$	Topic distribution over the microblog d
Λ	Label set for each microblog
Φ	Hyperparameters for Λ
z	Topic of word
w	Word in microblog
φ	Word distribution over topics
β	Hyperparameter for φ

Following the prior classification method in [10], we assume that the topic of a microblog is chosen from the subset of those labels, and each word w in microblog d is generated from a word distribution associated to one of that microblog's labels. In this way, the non-content words would be filtered out. To overcome the short text limitaion, we assume that the topic distribution of a microblog has strong relationships with the user's whole microblog set. For each user u , a Dirichlet distribution with K latent topics is modeled as

$$\theta_u = (\theta_{u,1}, \dots, \theta_{u,K})^T \sim Dir(\cdot | \alpha_u). \quad (4)$$

For each microblog, if it has @ relationships with other users, we believe the microblog topic is related to this set of neighbors' microblogs. The topic distributions of the neighbors is modeled as

$$\theta_c = (\theta_{c,1}, \dots, \theta_{c,K})^T \sim Dir(\cdot | \alpha_c). \quad (5)$$

After these analyses, $\theta^{(d)}$ for each microblog is generated by a linear combination of the aspects as

$$\theta^{(d)} = \mu_1 * \theta_d + \mu_2 * \theta_u + \mu_3 * \theta_c, \quad (6)$$

with the constraint that $\mu_1 + \mu_2 + \mu_3 = 1$.

The aspect of noise filtering is conducted as a prior stage before. Thus it does not occur in this combination. A normalization is conducted after the combination.

Based on the $\theta^{(d)}$, we suppose that the microblog has N_d words. And each word i in $\{1, \dots, N_d\}$ is generated in a similar way to the LDA model.

Specifically, draw a topic z first from the topic mixture over the microblog labels

$$z_i \sim Multi(\cdot | \theta^{(d)}), \quad (7)$$

and then draw a word w_i from the z_i topic-word distribution

$$w_i \in \{1, \dots, N\} \sim Multi(\cdot | \varphi_{ki}). \quad (8)$$

4 Estimating the Parameters

The topic for the word is based on the distribution of all known and hidden variables given the hyperparameters as

$$P(w, z | \alpha_d, \alpha_u, \alpha_c, \beta, \Phi, \lambda) = P(w, z | \lambda, \theta, \varphi) P(\theta | \alpha_d, \alpha_c, \alpha_u, \Phi) P(\varphi | \beta). \quad (9)$$

The topic distribution for a microblog is calculated by a linear combination of three parts as

$$P(\theta | \alpha_d, \alpha_u, \alpha_c, \Lambda) = \mu_1 * P(\theta_d | \alpha_d) + \mu_2 * P(\theta_u | \alpha_u) + \mu_3 * P(\theta_c | \alpha_c) \quad (10)$$

We use Gibbs sampling to train the model.

For microblog d , the posterior probability for its topic assignment would be

$$P(\theta_d | \alpha_d) = \frac{n_{k,-i}^d + \alpha_k^d}{\sum_{k=1}^K (n_{k,-i}^d + \alpha_k^d)}, \quad (11)$$

where $n_{k,-i}^d$ is the number of words (except the i^{th} word) in microblog d that are assigned to topic k .

Similarly, for user u , the posterior probability for its topic assignment would be

$$P(\theta_u | \alpha_u) = \frac{n_{k,-i}^u + \alpha_k^u}{\sum_{k=1}^K (n_{k,-i}^u + \alpha_k^u)}, \quad (12)$$

where $n_{k,-i}^u$ is the number of words (except the i^{th} word) in user u that are assigned to topic k .

The topic distribution from the social network aspects is calculated as

$$P(\theta_c | \alpha_c) = \frac{n_{k,-i}^c + \alpha_k^c}{\sum_{k=1}^K (n_{k,-i}^c + \alpha_k^c)}. \quad (13)$$

Therefore, for the i^{th} token in the d^{th} microblog for user u , the conditional posterior for its latent topic would be

$$\begin{aligned} P(z_i = k | z_{-i}) &\propto (\mu_1 * \frac{n_{k,-i}^d + \alpha_k^d}{\sum_{k=1}^K (n_{k,-i}^d + \alpha_k^d)} + \mu_2 * \frac{n_{k,-i}^u + \alpha_k^u}{\sum_{k=1}^K (n_{k,-i}^u + \alpha_k^u)} \\ &+ \mu_3 * \frac{n_{k,-i}^c + \alpha_k^c}{\sum_{k=1}^K (n_{k,-i}^c + \alpha_k^c)}) * \frac{n_{k,-i}^{w_i} + \beta_{w_i}}{\sum_{i=1}^N (n_{k,-i}^{w_i} + \beta_{w_i})}. \end{aligned} \quad (14)$$

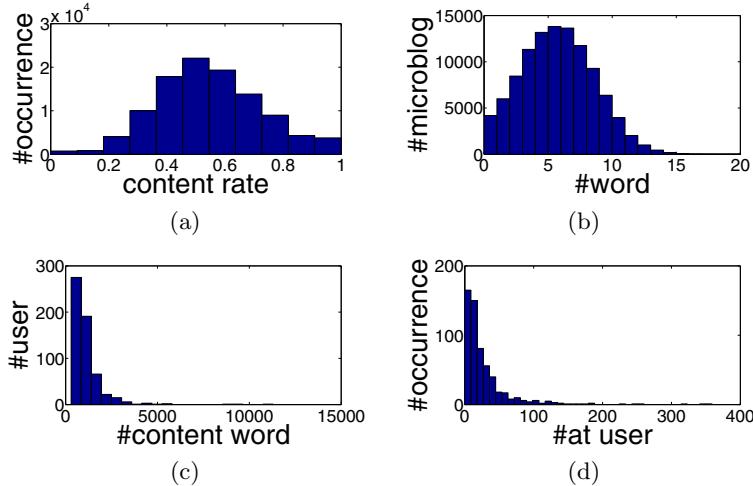


Fig. 4. The statistics for the dataset

After the sampling process, the parameters are estimated as

$$\varphi_{k,i} = \frac{n_k^{w_i} + \beta_{w_i}}{\sum_{i=1}^N (n_k^{w_i} + \beta_{w_i})}, \quad (15)$$

$$\theta_{d,k} = \frac{n_k^d + \alpha_k^d}{\sum_{k=1}^K (n_k^d + \alpha_k^d)}, \quad (16)$$

$$\theta_{u,k} = \frac{n_k^u + \alpha_k^u}{\sum_{k=1}^K (n_k^u + \alpha_k^u)}. \quad (17)$$

5 Experiments

In this section, we make the experimental verification on the performance of the proposed approach. Both qualitative analysis and quantitative analysis are conducted. In the qualitative analysis, we utilize the same example in Fig. 1 to demonstrate that the proposed model have all the advantages of the previous methods intuitively. In the quantitative analysis, we evaluate different methods under the metrics of log-likelihood and perplexity. We show that the proposed approach could consistently outperform the previous methods at a significant scale.

5.1 Experimental Setup

We utilize the SNAP [8,13] dataset from Stanford as our dataset. This dataset is collected from Twitter, from June 1st 2009 to June 30th 2009. The original

Method	Topics	probability
Combination-LDA	24 crime,people,Tehran,demonstration,mousavi,election,hold,right	0.5162
	39 political,public,guide,destroy,crime, chief,financial,officer	0.0586
	38 photo,military,soldier,kick,wars,potent,tougher,sleep	0.0349
Social-LDA	35 first,danger,mousavi,election, demonstration,crime,support,just	0.1923
	21 right,join,mousavi, opposition, president ,vote,political	0.1832
	2 people,police,protect,day,fear,home,destroy,crime	0.0676
User-LDA	20 Iran,tehran,today, demonstration, media,mousavi,police,election	0.2359
	23 Iran,good, Iranian,social,free,lost,opposition,confirmed	0.2092
	48 life,time,police,health,report,women,just,join,hlep	0.1477
Content-LDA	32 Iran,people,Tehran,news,iranian,world,mousavi, demonstration	0.1917
	11 day,video, live, probably, post, social, via, follow	0.1025
	9 Iran,tracker,welcome,Tehran,run,song,search,listen	0.0801
Original-LDA	5 just,like,people,time,Tehran,help,crime,follow	0.0996
	7 Iran,love,day,thanks,time,think,please,see	0.0831
	19 Iran,will, H...,obama,first,video,police,show	0.0819

Fig. 5. A qualitative example for characterizing microblogs' topics

dataset is in large scale. Following the ideas in [10], a subset is extracted for the calculation convenience. We retain the most active users in the extracting subset. For the original dataset, an approximate optimization approach could be utilized in the collapsed Gibbs-sampling [15], which could obtain competitive results from the precise approach. The subset contains 585 users with 107,519 microblogs.

Some statistics are shown in Fig. 4. To show how noisy the microblogs' texts are, we divide the words in each microblog into content words and noise words. The noise words include the stopword, the symbol, and etc. The rate of content words in each microblog is calculated, and the distribution is drawn in Fig. 4 (a). It follows the Gaussian distribution. From the figure, it could be obtained that in most cases, 50% of the words in each microblog are noise words, which is in a large percent. To show how short the microblog's length is, we draw the distributions of microblogs' length in Fig. 4 (b). It follows the Gaussian distribution. The average number of words is only 5. We also draw the number of content word in each user's blog history as shown in Fig. 4 (c). It follows the power-law distribution, indicating that most users have only a few content words. Finally, we also draw the distribution of how many friends each user @ in history in Fig. 4 (d). It also follows the power-law distribution. Most users have linked less than 50 friends.

We choose four previous methods for comparisons, including the original LDA model [1], the Content-LDA model [10], the User-LDA model [15], and the Social-LDA model [14]. In the configuration of filtering noise words, following the work in [10], we set three labels, including the content word, the stopword, and symbols (URLs, emoticons, punctuations, and hashtags).

5.2 Qualitative Analysis

Fig. 5 shows the qualitative analysis of the proposed approach. The results come out as we expected in the first section. Due to the noise, the short text and

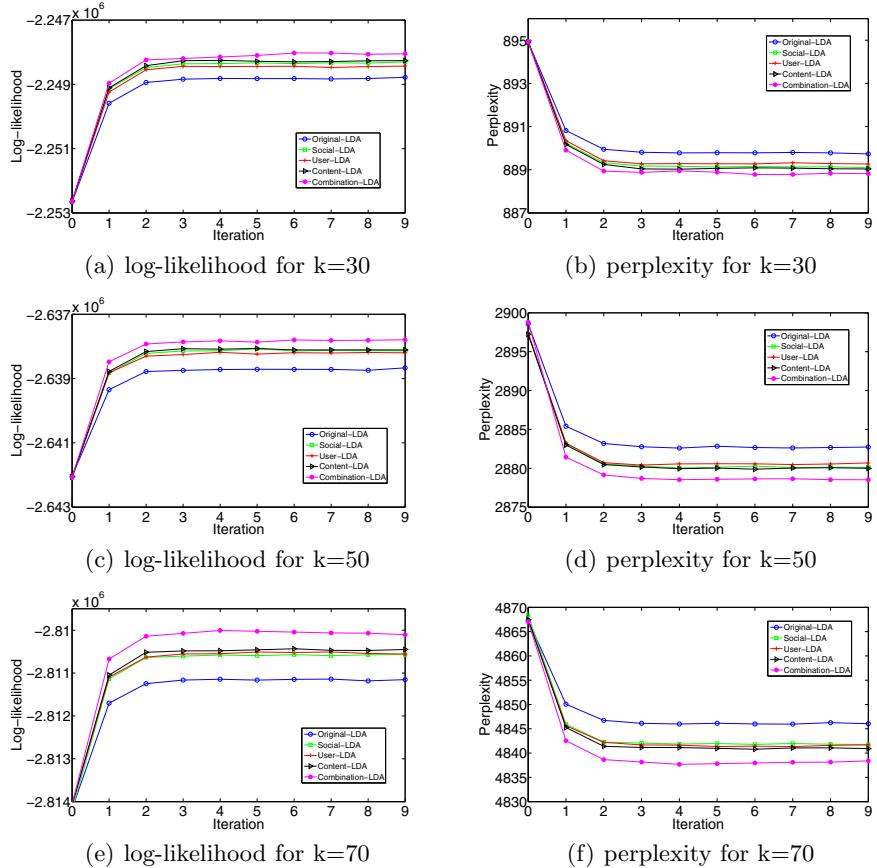


Fig. 6. Quantitative analysis for the proposed approach

content incompleteness, the original LDA model performs the worst. The generated topics have a lot of noise, and the topics are not accurately extracted and apparently assigned. Although the Content-LDA, the User-LDA and the Social-LDA overcome some of the limitations separately, the information are not sufficiently utilized. Therefore, it is hard for the model to choose from the top two or three topics. The proposed combination model performs the best. It maps the microblog precisely to the topic of the demonstration event happened during the Iranian election. This example demonstrates intuitively the effectiveness of the combination.

5.3 Quantitative Analysis

In the quantitative analysis, we utilize the log-likelihood and perplexity to evaluate the effectiveness of the models in characterizing microblogs' topics. The detailed definition of the metrics could be found in [4]. For the log-likelihood, the

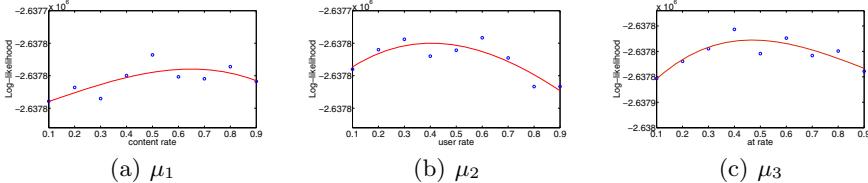


Fig. 7. Sensitivity analysis for the parameters

larger the value, the better the performance; and for the perplexity, the smaller the value, the better the performance. We conduct the experiments in different configurations. The latent topic number is set to 30, 50, and 70, respectively.

Fig. 6 shows the quantitative analysis results. It could be obtained that in all the configurations, the proposed model consistently outperforms the other methods at a significant scale. This demonstrates the effectiveness of the proposed methods. It overcomes all the three limitations of microblogs, including noise, short text, and content incompleteness. The Content-LDA, the User-LDA, and the Social-LDA overcome one limitation each, thus they outperform the original LDA.

We also conduct the sensitivity analysis for the parameters of μ_1 , μ_2 , and μ_3 in Eq. 14. The result is shown in Fig. 7. In Fig. 7 (a), the proportion of microblog content is range from 0.1 to 0.9, and the other two parameters divide the left proportion equally. Similarly, Fig. 7 (b) shows the result of changing the proportion of user topic distribution, and Fig. 7 (c) illustrates the result of changing the proportion of @user. It could be observed that there is a peak in all the figures. Both large value and small value would reduce the performance. The model is not sensitive to the parameters when they are between 0.3 and 0.7.

6 Conclusion

In this paper, we propose a unified generative model for characterizing microblogs' topics. It combines the advantages of three previous models. We also derive a collapsed Gibbs-sampling optimization method for estimating the parameters. Through both qualitative and quantitative analysis in a real world dataset from Twitter, we demonstrate that the proposed approach can effectively overcome the three challenges of modeling microblogs, including 1) too much noise, 2) short text, and 3) content incompleteness.

Acknowledgments. The work described in this paper was fully supported by the National Basic Research Program of China (973 Program, Grant No. 2013CB329605). The authors also would like to thank the reviewers for their helpful comments.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *The Journal of Machine Learning Research* 3, 993–1022 (2003)
2. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.P.: Measuring user influence in twitter: The million follower fallacy. In: 4th International AAAI Conference on Weblogs and Social Media (ICWSM), vol. 14, p. 8 (2010)
3. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41(6), 391–407 (1990)
4. Heinrich, G.: Parameter estimation for text analysis 2(1), 4–3 (2005), Web: <http://www.arbylon.net/publications/textest>
5. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57. ACM (1999)
6. Hong, L., Ahmed, A., Gurumurthy, S., Smola, A.J., Tsoutsouliklis, K.: Discovering geographical topics in the twitter stream. In: Proceedings of the 21st International Conference on World Wide Web, pp. 769–778. ACM (2012)
7. Java, A., Song, X., Finin, T., Tseng, B.: Why we twitter: understanding microblogging usage and communities. In: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis, pp. 56–65. ACM (2007)
8. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: WWW 2010: Proceedings of the 19th International Conference on World Wide Web, pp. 591–600. ACM, New York (2010)
9. Lin, C.X., Mei, Q., Han, J., Jiang, Y., Danilevsky, M.: The joint inference of topic diffusion and evolution in social communities. In: 2011 IEEE 11th International Conference on Data Mining (ICDM), pp. 378–387. IEEE (2011)
10. Ramage, D., Dumais, S., Liebling, D.: Characterizing microblogs with topic models. In: International AAAI Conference on Weblogs and Social Media, vol. 5, pp. 130–137 (2010)
11. Ramage, D., Hall, D., Nallapati, R., Manning, C.D.: Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, vol. 1, pp. 248–256. Association for Computational Linguistics (2009)
12. Tumasjan, A., Sprenger, T.O., Sandner, P.G., Welpe, I.M.: Predicting elections with twitter: What 140 characters reveal about political sentiment. In: Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media, pp. 178–185 (2010)
13. Yang, J., Leskovec, J.: Patterns of temporal variation in online media. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 177–186. ACM (2011)
14. Zhang, C., Sun, J., Ding, Y.: Topic mining for microblog based on mb-lda model. *Journal of Computer Research and Development* 48(10), 1795–1802 (2011)
15. Zhao, X., Jiang, J.: An empirical comparison of topics in twitter and traditional media. Singapore Management University School of Information Systems Technical Paper Series 10, 2011 (retrieved November 2011)

A Novel Model for Medical Image Similarity Retrieval

Pengyuan Li¹, Haiwei Pan^{1,*}, Jianzhong Li², Qilong Han¹,
Xiaoqin Xie¹, and Zhiqiang Zhang¹

¹ College of Computer Science and Technology, Harbin Engineering University, Harbin, China

² School of Computer Science and Technology, Harbin Institute of Technology, Harbin China

pengyuanli@yahoo.cn, heaven_007cn@yahoo.com.cn

Abstract. Finding the similar medical images from medical image database can help doctors diagnose based on the cases before. However the similarity retrieval for medical images requires much higher accuracy than the general images. In this paper, a new model of uncertain location graph is presented for medical image modeling and similarity retrieval. Then a scheme for uncertain location graph retrieval is introduced. Furthermore, an index structure is applied to reduce the searching time. Experimental results reveal that our method functions well on medical images similarity retrieval with higher accuracy and efficiency.

Keywords: uncertain location graph, image modeling, medical image.

1 Introduction

Over the last decades, the medical imaging technology such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) help doctors to diagnose diseases with the medical images. Hence, a vast amount of medical images are generated from hospitals every year. By using image-to-image comparison doctors will find out the similar images which come from different patients. These patients may have high possibility to get same disease because they have the similar pathological characteristics in their images. According to this domain knowledge, we believe that finding the similar images from medical image database will significantly assist doctors to find patients who may get the same disease. And it will also help doctors to make diagnoses by acquiring information from the previous diagnoses and results.

Up to now, there are two categories of image retrieval methods: 1) description-based image retrieval, which is performed object retrieval based on image descriptions. 2) content-based image retrieval (CBIR), which supports retrieval based on the image content [1]. Due to the image always has complicated semantic information and it is hard to describe an image with limited words, CBIR is an increasingly popular discipline in computer science [2]. Some simple and elegant methods were developed for

* Corresponding author.

CBIR. [3] proposed a method based on relevance feedback, [4] introduced a method based on medical image sequence, and [5] proposed a method by wavelet-based CBIR.

But medical image similarity retrieval is different from general image similarity retrieval [6]. Texture is an important spatial feature for the doctors' diagnosis because the texture can describe the objects naturally and even present the gray value variation of adjacent pixels. [7] used LBP and spatial indexing for medical image retrieval. And several researches [8] focused on shape description and matching problem. But most texture descriptor is static value which is calculated by some texture features. It has less consideration about the uncertainty and structure of the texture.

At the same time, graph mining is a growing research topic because more and more objects can naturally modeled by graph [9,10]. A novel model of uncertain graph [11] enlarges the application of graph model for the practical objects. But this model which is a kind of probability graph cannot be applied directly to the image. To our best knowledge there is no research on medical image retrieval with uncertain graph.

In this paper, we firstly propose a novel model of uncertain location graph. It is a kind of uncertain graph based on uncertainty of vertex's location. Secondly, a new method is introduced to model brain CT images to uncertain location graphs based on image texture. And a novel concept of texel which is the fundamental of texture is applied to maintain the structure of texture. Then an approximate uncertain location graph matching scheme is presented for medical image similarity retrieval. Furthermore, we propose an index structure to speed up the medical image retrieval process.

This paper is organized as follows: Section 2 introduces a preprocessing work. Section 3 proposes the model of uncertain location graph and gives a method for modeling medical images to uncertain location graphs. Section 4 describes the uncertain location graph similarity retrieval method. Extensive experiments are shown in Section 5. And section 6 concludes the paper and highlights the future works.

2 Preprocessing

Through the observation of a large number of brain CT images, we find out that the brain CT image has the following advantages: 1) The size of brain in the brain CT images are relatively stable. 2) The texture location of the brain CT image always has the same semantic. 3) The brain CT image has a simple background. Fig. 1 (a) is an original brain CT image. According to our survey doctors, we know that the hypodense (dark) structures indicate much more information than hyperdense (bright) structures. That also means the texture of hypodense structures is more important than the texture of hyperdense structures during the texture compare process. With this domain knowledge, we give different part of texture different value based on gray histogram to express the different importance of the different part of texture. Therefore, the preprocessing transforms the brain CT image to uniformed hierarchical texture image and it is the basic to transform brain CT image to uncertain location graph.

Every brain CT image in the database contains $m \times n$ pixels corresponding a matrix $P_{m \times n}$. The value of $P(i, j)$ is the gray value of the j th column of the i th row pixel in the image. The value of $P(i, j)$ smaller, the pixel from the image located i th column of the j th row darker. The scheme of the preprocessing is: 1) Extract the region of interest of the image like Fig. 1 (b). 2) Calculate the gray histogram of the region of interest and

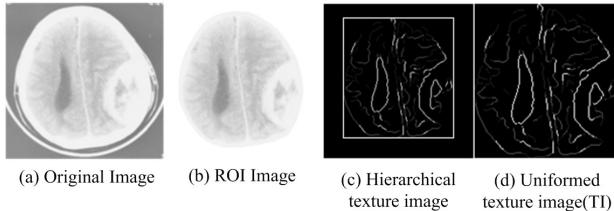


Fig. 1. The process of preprocessing

find the k layers partition $\text{par}[k]$. 3) With the canny edge detector [12] a hierarchical texture image is generated. Fig. 1 (c) is an example of the hierarchical texture image. The output of the preprocessing is the uniformed texture image TI with the uniform size COLUMN×ROW like Fig. 1 (d). In this paper COLUMN=161, ROW=151.

With this preprocessing, every original image in medical image database can be transformed to a uniformed texture image TI and it corresponds to a matrix TM. TM(i,j) is the gray value of the jth column of the ith row pixel in TI and $\forall TM(i, j) \neq 0$ in the image means there is a texture through the location (i, j).

3 Medical Image to Uncertain Location Graph

In this section, we firstly give the definition of uncertain location graph. Secondly, a modeling method is proposed. And a concept of texel is applied to reduce the complexity of uncertain location graph. Thirdly, we describe the texel mobility which based on vertex mobility and structure of the texel.

3.1 Uncertain Location Graph

As usual, we assume that the graph is an abstract representation with the static vertex set V and the edge set E. Uncertain location graph is a kind of undirected graph $G=(V, E)$ which every vertex in the graph has a uncertain location. Every uncertain location graph is in a coordinate system, and we just display the coordinate system in Fig 2.

Definition 1. An uncertain location graph (ULG) is a system $G=(V, E, L, P, T)$, where V is the vertex set, E is the edge set, $L: V \rightarrow \{1, 2, 3, \dots, n\}$ is a function assigning labels to vertices, $P: V \rightarrow [0, 1]$ is a function assigning importance values to vertices, $T=\{T(1), T(2), \dots, T(m)\}$ and $T(i)$ is a subset of V .

In this paper, $V(i)$ is a vertex of the graph with the label i by L functions, and $V(i)$ has the location or the relative location (x, y). $P(V(i))$ is the importance value of the vertex $V(i)$ which means that if the vertex $V(i)$ moves it will cause $P(V(i))$ impact on the graph. The maximum $P(V(i))=1$ means if $V(i)$ moves little, the graph is completely different. The minimum $P(V(i))=0$ means it has no impact to the graph no matter how far $V(i)$ moves. As we can see, the $P(V(i))$ also means the mobility of $V(i)$, low importance means high mobility and vice versa. So, we define the mobility of $V(i)$ as:

$$m(V(i)) = 1/P(V(i)) \quad (1)$$

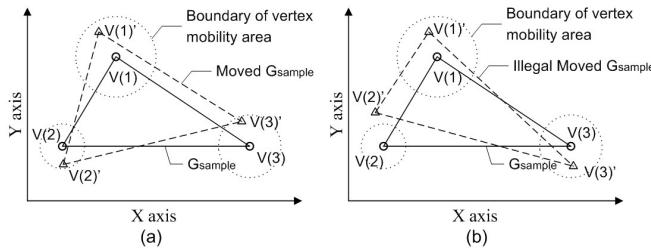


Fig. 2. The uncertainty of uncertain location graph

Since the vertex in the ULG has mobility, the edge in the ULG is also not static. The edge $(V(i), V(j))$ between $V(i)$ and $V(j)$ can move by $m(V(i))$ and $m(V(j))$ limited.

Example 1. There is an ULG $G_{\text{sample}} = \{ \{V(1), V(2), V(3)\}, \{(V(1), V(2)), (V(2), V(3)), (V(1), V(3))\}, L, P, T = \{T(1), T(2)\} \}$, where $T(1) = \{V(1), V(2)\}$, $T(2) = \{V(3)\}$ in Fig 2(a). Every vertex has different importance value $P(V(2)) > P(V(3)) > P(V(1))$ and with formula 1 we can know $m(V(2)) < m(V(3)) < m(V(1))$. The circle around the vertex is the boundary of vertex mobility area which the vertex is the center and the mobility m of the vertex is the radius. The uncertainty of ULG is mainly about the vertex of the graph can move by their mobility area constraint. $V(1)', V(2)', V(3)'$ are the location that vertex $V(1)$, $V(2)$, $V(3)$ may moved, and we consider it is also graph G_{sample} even though the vertex has been moved. Fig 2.(b) shows an illegal move of ULG G_{sample} because the location of $V(2)'$ is out of vertex $V(2)'$'s mobility area.

3.2 Modeling Method

After preprocessing we have got the uniformed texture image TI and it corresponds a matrix TM. There are five steps to transform a uniformed texture image TI to an ULG.

Vertex. $TM(i, j)$ is the gray value of the j th column of the i th row pixel in the TI, and $\forall TM(i, j) \neq 0$ in the image TI means there is a texture through the location (i, j) . So we define there is a vertex with location (i, j) in the ULG.

Edge. There is no distance of any pair of adjacent pixels larger than $\sqrt{2}$. Therefore, we give the definition of edge: $\{(i, j), (k, l) \in E \mid TM(i, j) \neq 0, TM(k, l) \neq 0, \text{dis}((i, j), (k, l)) \leq \sqrt{2}\}$.

P. The preprocessing gives the different value to $TM(i, j)$ based on their gray value and with the domain knowledge it is also an importance value of the vertex (i, j) . Therefore the function P is:

$$P(i, j) = TM(i, j) \quad (2)$$

Definition 2. A texel $T(k) = \{V(a), V(i), V(j), \dots, V(b)\}$ is a path with no repeated vertices from $V(a)$ to $V(b)$ in the uncertain location graph. Where $\forall i, j, T(i) \cap T(j) = \emptyset$, and $V = \{T(1) \cup T(2) \cup \dots \cup T(m)\}$.

For image retrieval, texture feature is always too complex to be described, and feature point is always too simple to represent the total texture feature. In this paper we describe the texture feature with texels. With the definition of texel we know that the texel is the fundamental element of texture. It consists of a sequence of vertices and maintains the structure of sectional texture.

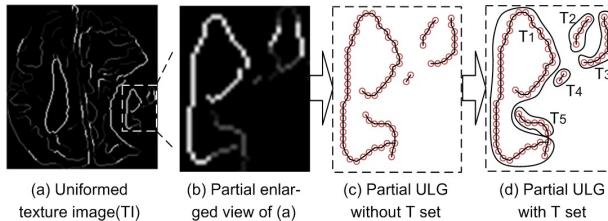


Fig. 3. An example of texture image to ULG

T. T is a texel set which $T = \{T(1), T(2), \dots, T(m)\}$. T is the segmentation of ULG. At the same time, T is the segmentation of the texture.

For a texel the generating process is: 1) Find a start-point; 2) Find a vertex which has a edge with last vertex and it has never been labeled. If the vertex has more than one vertex connected and unlabeled we choose the vertex according the rule: 1. Choose the vertex which importance value is similar to the last one; 2. If the vertices have the same importance value, we choose the vertex randomly; 3) Execute step 2 until there is no vertex can be found. One segmentation result is shown in Fig. 3.(d).

L. Function L labels the vertex in order during the generating T process.

3.3 Texel Mobility

Since the vertex in the ULG has the mobility, the texel which consists of a sequence of vertices also has the mobility that we called *texel tolerance*. As Fig. 2 shown, the mobility of $V(i)$ depends on $P(V(i))$, the texel tolerance depends on the importance value as well. For each vertex on the texel, there is an edge to the next vertex. The basic area that texel can be moved is formulated by the vertex mobility area. Therefore, we define the basic flexibility of texel tolerance standard b_{tt} as:

$$b_{tt}(T(k)) = \sum m(V(i))/n, V(i) \in T(k) \quad (3)$$

which n is the number of vertices in $T(k)$. It means the vertex on the texel can be moved by their texel tolerance standard b_{tt} constraint. Two examples are given in Fig. 4 (a), texel a is the original texel, texel b and c are the texels that texel a might be moved. The circle in Fig. 4 (a) is the boundary of vertex mobility area which the b_{tt} is the radius. Nevertheless, when a part of texel moved merely and the other parts of the texel moved farther than b_{tt} we also recon it is OK. Weber-Fechner law talked that subjective sensation is proportional to the logarithm of the stimulus intensity [13], which means if a stimulus varies as geometric progression, the corresponding perception will altered in an arithmetic progression. Because the appearance of texel

is irregular, we take the length of texel as the stimulus and 2 as the common ratio of the geometric progression. Then the extend texel tolerance standard e_{tt} defined as:

$$e_{tt} = \begin{cases} b_{tt}, & \log_2 \text{length} \leq b_{tt} \\ \log_2 \text{length}, & \log_2 \text{length} > b_{tt} \end{cases} \quad (4)$$

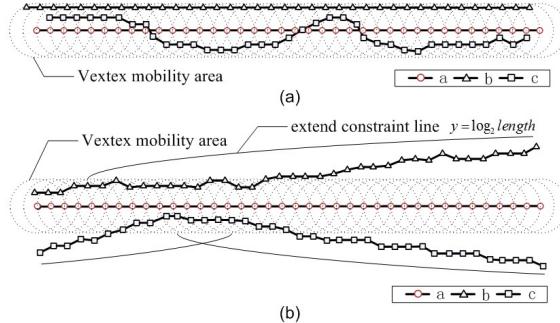


Fig. 4. The flexibility of texel in uncertain location graph

Where the length is the hop from the vertex which is going to be moved to the longest vertex which inside it's mobility area. Fig. 4 (b) gives two examples that texel b and c are the texels that texel a may moved with the e_{tt} . From these examples we can find that the length we mentioned in the formula 4 is not static, and the e_{tt} is depends on the variability of the sequence of vertices and vertex's sequence number. That also means the texel in ULG has dynamic mobility which based on the structure of texel.

4 ULG Similarity Retrieval Problem

In this section, we firstly give the definition of similarity between ULGs. Then a basic scheme for ULG similarity retrieval ULGR is introduced. To speed up the retrieval process an index is applied and an advanced ULG similarity retrieval method with an index ULGR-index is proposed. In this paper, we denote $G=\{V, E, L, P, T\}$ as a query ULG, $G'=\{V', E', L', P', T'\}$ is an ULG of the ULG set G_{db} which will be searched.

4.1 Similarity between ULGs

In this subsection, two levels of correspondence will be defined: matching and similar. With these following definitions the similarity between two ULGs can be calculated. And a corollary will be proposed for future ULG similarity retrieval.

Definition 3. Vertex $V(i)$ and $V'(j)$ is matching if $\text{dis}(V(i), V'(j)) \leq m(V(i)) + m(V'(j))$.

Definition 4. Texel $T(i)$ and $T'(j)$ is matching if every vertex $V(k)$ on the vertex $T(i)$ has a correspondence to the vertex $V'(l)$ on the texel $T'(j)$, and every pair of vertices $V(k)$ and $V'(l)$ satisfy $\text{dis}(V(k), V'(l)) \leq b_{tt}(T(i)) + b_{tt}(T'(j))$.

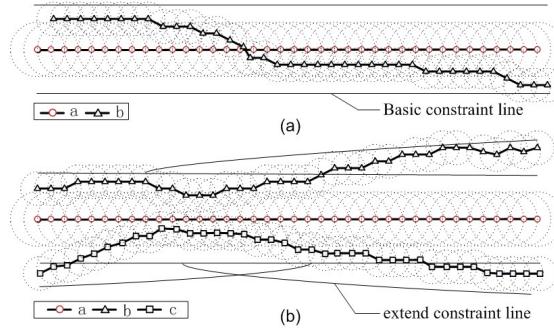


Fig. 5. The similarity between two texels

Fig 5 (a) gives an example that *texel b* matching *texel a*. Because every vertex on *texel b* has a correspondence to vertex on *texel a* and satisfies the constraint. The basic constraint line is a line with the distance= $b_{_tt}(T(i))+b_{_tt}(T'(j))$ from *texel a*.

Definition 5. *Texel T'(j) is similar to T(i) if every vertex V(k) on the texel T(i) has a correspondence to the vertex V'(l) on the texel T'(j) and satisfies the constraint $dis(V(k), V'(l)) \leq b_{_tt}(T(i))+b_{_tt}(T'(j))+\log_2 length$.*

Where the length is the hop from the vertex $V'(l)$ to $V'(m)$ which $V'(m)$ is the farthest vertex has been matched. Fig.5 (b) gives an example of *texel b, c* similar to *texel a*. Extend constraint line is a line which has the distance= $b_{_tt}(T(i))+b_{_tt}(T'(j))+\log_2 length$ from *texel a*.

Definition 6. *Texel T'(j) is partially similar to T(i) if a part of vertices V(k) on texel T(i) have correspondence to the vertices V'(l) on texel T'(j) and satisfy $dis(V(k), V'(l)) \leq b_{_tt}(T(i))+b_{_tt}(T'(j))+\log_2 length$.*

With these definitions, the similarity between two texels *st* can be calculated as:

$$st(T(i), T'(j)) = \frac{\sum_{V(l) \in PT(i)} P(V(l))}{\sum_{V(k) \in T(i)} P(V(k))} \quad (5)$$

$PT(i)$ is the vertex set which consists of the vertices have been satisfied the constraint.

Definition 7. *The similarity sg of ULG G' to ULG G can be calculated by below:*

$$\begin{aligned} sg(G', G) &= \frac{\sum_{T(i) \in T} (st(T(i), T'(j)) * \sum_{V(m) \in T(i)} P(V(m)))}{\sum_{T(i) \in T} \sum_{V(m) \in T(i)} P(V(m))} \\ &= \frac{\sum_{T(i) \in T} st(T(i), T'(j))}{n} \end{aligned} \quad (6)$$

where n is the number of texels in T .

Definition 7 gives the definition of similarity between two ULGs. It pays more attention to the important texel which contains more important vertex or the longer one.

Corollary. If a pair of vertices $V(i)$ and $V'(k)$ is matched and another pair of vertices $V(j)$ and $V'(l)$ satisfy the constraint $\text{dis}(V(j), V'(l)) \leq b_{\text{tt}}(T(i)) + b_{\text{tt}}(T'(j)) + \log_2 \text{length}_{\{V(i), V(j) \in T(m), V'(k), V'(l) \in T'(n)\}}$, where the length is the hop from $V(i)$ to $V(j)$, we guess texel $T'(n)$ is similar or partial similar to $T(m)$. And we call $V(i), V'(k)$ are start-points and $V(j), V'(l)$ are end-points.

4.2 ULG Similarity Retrieval Method (ULGR)

In classical graph theory, graph matching is a NP-complete problem. Algorithm 1 gives a basic scheme for ULG similarity retrieval. It compares each texel in the query graph with every texel in an ULG which in the ULG database. Suppose that there is a query ULG G with m texels, and there are n ULGs with average t texels in G_{db} , there need $O(m * n * t)$ to compare the texels. If we assume every texel has the same number of vertices l , there need $O(m * n * t * l * l / 2)$.

The main problem of Algorithm 1 is time-consuming to compare the impossible similar texels and for each pair of potential texels it needs $O(l * l * l / 2)$ in average. The ideal method is to just compare the similar texels with $T(i)$ which is a texel in query ULG. Therefore, the problem is how to find texels which are possible similar to $T(i)$.

Algorithm 1. Basic ULG similarity retrieval method (ULGR)

Input: query ULG G , ULG database G_{db} .

Output: rank

Begin

```

    for each uncertain location graph  $G_z$  in  $G_{\text{db}}$ 
        for each  $T(i) \in G$ 
            for each  $T_z(j) \in G_z$ 
                for each pair of vertices  $V(m), V(n) \in T(i)$  where
 $V(m) \neq V(n)$ 
                    for each pair of vertices  $V_z(p), V_z(q) \in T_z(j)$  where
 $V_z(p) \neq V_z(q)$ 
                        calculate the similarity  $st(T(i), T_z(j))$ 
                    End for
                End for
            End for
        End for
    calculate  $sg[z] = sg(G_z, G)$ 
End for
rank the sg
End
```

4.3 The Index for ULG Similarity Retrieval

The best improvement must be finding the texel set T_{wl} which consists of the texels that may similar to texel $T(i)$ directly. In this paper every ULG is in a Column*Row plane. With the corollary, we design the index shown in Fig. 6 to find T_{wl} which consists of the texels that may be partially similar to texel $T(i)$ from $V(b)$ to $V(y)$.

The size of Pixel Index in Fig 6 is COLUMN*ROW. Every point PI(i,j) located (i,j) in pixel index has a Trans Table TT(i,j) which records the texels have ever crossed the (i,j) location from different brain CT images. And every item in Trans Table TT(i,j) corresponds the texel information which it belongs to. The texel information is recorded in the Texture Table. With the corollary, Algorithm 2 is proposed to find T_wl. The main idea of Algorithm 2 is to find the pair of vertices V(bb) and V(yy) which satisfy the constraints talked in corollary.

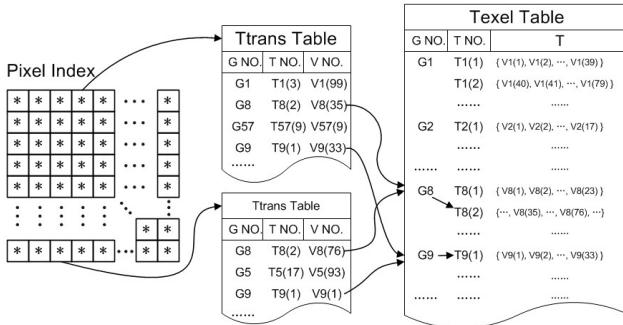


Fig. 6. The index for medical image similarity search

Algorithm 2. Find texel set T_wl that may partial similar to T(i) from V(b) to V(y)

Input: query ULG G, ULG database G_db

Output: T_wl[]

Begin

```

Initialize T_wl[], wl^[], wl^[], length=y-a;
wl^[] = U {TT(i, j) | dis((i, j), V(b)) <= b_tt(T(i)) }
wl^[] = U {TT(i, j) | dis((i, j), V(b)) <= b_tt(T(i)) + log2length}
T_wl[]' = wl^[] ∩ wl^[]
wl^[]' = U {TT(i, j) | dis((i, j), V(y)) <= b_tt(T(i)) }
wl^[]' = U {TT(i, j) | dis((i, j), V(y)) <= b_tt(T(i)) + log2length}
T_wl[]'' = wl^[]' ∩ wl^[]
T_wl[] = T_wl[]' ∪ T_wl[]''

```

End

With the index, advanced method ULG similarity retrieval with index (ULGR-index) is described as follows: 1)For each texel T(i) in G, find a pair of vertices as the start-point and end-point; 2)Find the texel set T_wl which consists of the texels that may be similar to T(i); 3)For each texel T(j) in T_wl, calculate st(T(i),T(j)) and write it into scorelist[m][n] which m is the number of texel in T, n is the number of ULG in database. 4) Execute step 2 until cannot find a pair of vertex as the start-point or end-point. 5) Execute step 1 until all texels have been calculated. 6) With the scorelist[], compute the similarity score for each ULG by formula 6 and give the rank. ULGR-index just need O(m*l*p*l/4*b_tt), which p is the number of the texel that may be similar to texel T(i) and b_tt is the average of all b_tt(T(i)) in G.

5 Experiments

In this section, three experiments were implemented to demonstrate and compare the performance of our methods for medical image retrieval. For comparison, two methods we proposed above ULGR, ULGR-index and other three algorithms were also implemented, KLT [14], LBP and Sp-LBP. KLT is a feature tracker, and evaluates the consistency of features between non-consecutive frames. LBP is a local texture descriptor with the grayscale invariant. Sp-LBP is the LBP approach with spatial indexing. In our experiments we use 8 pixels as a symmetric neighbor set and the spatial indexing is with 3 annular splits and 8 angular splits. Our experiment data includes 2000 brain CT images. All experiments are implemented in C/C++ using Intel Core(2) processor 2.66GHz, and 2GB of RAM.

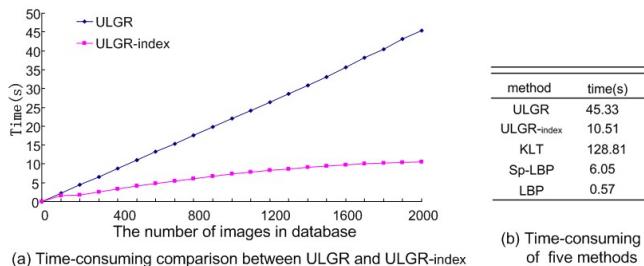


Fig. 7. The time-consuming comparison

5.1 Time-Consuming Analysis

Fig. 7(b) shows the time-consuming of five methods. Fig. 7(a) shows the time-consuming comparison between ULGR and ULGR-index. From Fig. 7(a) we can see ULGR-index is always faster than ULGR whatever in small database or the database with 2000 images. The time-consuming increasing speed of ULGR is about 2-3 seconds per 100 images constantly. The time-consuming increasing speed of ULGR-index is under 1 second per 100 images, and with the increase of database the time-consuming increasing speed is getting down.

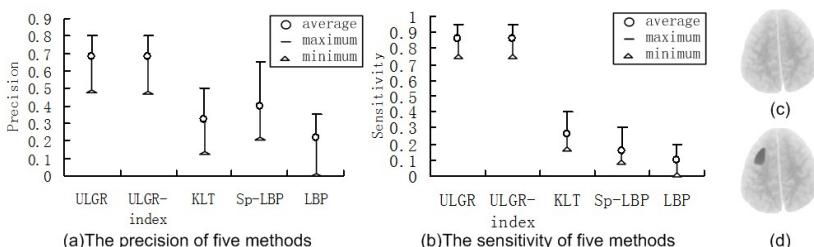


Fig. 8. The precision and sensitivity comparison of five methods

5.2 Precision Analysis

Performance of image similarity retrieval is generally measured by precision and recall, but for brain CT images we think each of them are relevant. In this experiment we simulated people just care about the top-20 results and we use precision(20) to measure the performance of each methods. Precision is calculated by:

$$\text{precision} = \frac{\text{number of relevant returns}}{\text{number of returns}} \quad (7)$$

Fig. 8(a) shows the precision of five different methods which is based on 50 independent queries. ULGR is almost same with ULGR-index in three parameters. And these two methods hold higher score in three parameters.

5.3 Sensitivity Analysis

Precision and recall are generally used to measure the performance of image similarity retrieval, but in the real world especially for medical images little change is crucial. Fig. 8 (c) is a brain CT image from a normal human being and Fig. 8 (d) is a image we put a simulated field in Fig. 8 (c). Although they look similar to each other, but doctors just focus on the pathology lesions. Therefore, we measure the sensitivity as:

$$\text{Sensitivity} = \frac{(S(a) - SC(a))}{S(a)} / S(a) \quad (8)$$

Where $S(a)$ is the relevant results of image a, and $SC(a)$ is the relevant results of little changed image a, like Fig. 8 (d), and we use the top-20 results as the relevant results.

From Fig. 8(b), one could observe ULGR and ULGR-index can distinguish the images which have a little change better than other three methods. That means the model of ULG can focus on the change of textures better especially for normal and abnormal brain CT images. And this characteristic is useful for diagnosis.

6 Conclusions

In this paper, a novel model of ULG is proposed. The uncertainty of ULG which is based on vertex's location is quite different with other models. And ULG is first used for medical image modeling and similarity retrieval. Experimental results show that our methods can find the image with similar texture more accurately. Secondly, our methods can distinguish the medical images which have little different from each other. And the efficiency index let ULGR-index has much advantage to ULGR in time complexity.

The model of ULG is fairly generic and can be extend to different applications. Future work may include other practical application with ULG model and reducing the ULG matching time.

Acknowledgment. The paper is partly supported by the National Natural Science Foundation of China under Grant No.61272184, 61202090, 61100007; Natural Science Foundation of Heilongjiang Province under Grant No.F200903, F201016, F201024,

F201130; The Program for New Century Excellent Talents in Universities(NCET-11-0829); The Fundamental Research Funds for the Central Universities under grant No.HEUCFZ1010, HEUCFT1202, HEUCF100609; The Science and Technology Innovation Talents Special Fund of Harbin under grant No. RC2010QN010024.

References

1. Jiawei, H., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn., pp. 396–399. China Machine Press, Beijing (2006)
2. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. *J. ACM Computing Surveys* 40(2), 5–60 (2008)
3. Wang, R., Pan, H., Han, Q., Gu, J., Li, P.: Medical Image Retrieval Method Based on Relevance Feedback. In: Zhou, S., Zhang, S., Karypis, G. (eds.) ADMA 2012. LNCS (LNAI), vol. 7713, pp. 650–662. Springer, Heidelberg (2012)
4. Pan, H., Han, Q., Xie, X., Wei, Z., Li, J.: A Similarity Retrieval Method in Brain Image Sequence Database. In: Alhajj, R., Gao, H., Li, X., Li, J., Zaïane, O.R. (eds.) ADMA 2007. LNCS (LNAI), vol. 4632, pp. 352–364. Springer, Heidelberg (2007)
5. Quellec, G., Lamard, M., Cazuguel, G., et al.: Fast Wavelet-Based Image Characterization for Highly Adaptive Image Retrieval. *IEEE Trans. on Image Processing* 21(4), 1613–1623 (2012)
6. Muller, H., Deserno, T.M.: Content-Based Medical Image Retrieval. In: Biomedical Image Processing, pp. 471–494. Springer, Heidelberg (2011)
7. Unay, D., Ekin, A., Jasinschi, R.S.: Local Structure-Based Region-of-Interest Retrieval in Brain MR Images. *IEEE Trans. on Information Technology in Biomedicine* 14(4), 897–903 (2010)
8. Shu, X., Wu, X.: A novel contour descriptor for 2D shape matching and its application to image retrieval. *Image and Vision Computing* 29(6), 286–294 (2011)
9. Caetano, T.S., McAuley, J.J., Cheng, L., et al.: Learning Graph Matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 31(6), 1048–1058 (2009)
10. Zhu, Y., Qin, L., Yu, J.X., et al.: High Efficiency and Quality: Large Graphs Matching. In: 20th ACM International Conference on Information and Knowledge Management, pp. 1755–1764. ACM, New York (2011)
11. Zou, Z., Li, J., Gao, H., et al.: Mining Frequent Subgraph Patterns from Uncertain Graph Data. *IEEE Trans. on Knowledge and Data Engineering* 22(9), 1203–1218 (2010)
12. Canny, J.: A Computational Approach to Edge Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 8(6), 679–698 (1986)
13. Reber, A.S.: The Penguin dictionary of psychology, 2nd edn. Penguin Press, New York (1995)
14. Shi, J., Tomasi, C.: Good features to track. In: The IEEE International Conference on Computer Vision and Pattern Recognition, pp. 593–600. IEEE Press, Seattle (1994)
15. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley Interscience, New York (1991)

Finding Similar Questions with Categorization Information and Dependency Syntactic Tree

Xin Lian, Xiaojie Yuan*, Xiangyu Hu, and Haiwei Zhang

Institute of Computer Science and Technology, Nankai University, Tianjin, China
{lianxin,yuanxiaojie,huxiangyu,zhanghaiwei}@dbis.nankai.edu.cn,
forwarding82@gmail.com

Abstract. Question Answering communities rapidly build up large archives of questions and answers. One of the major tasks in a question and answer service is to find similar questions to a new question. Question retrieval in the CQA sites is different from web search. It doesn't take advantage of the features of CQA sites to introduce natural language to solve the problem. In this paper, we address this problem by utilizing more features of CQA sites, including question, description, answer, category and users' posted questions. The model is divided into question classification and question retrieval. Question classification prunes the search space and removes some noise. Then "dependency syntactic tree" is made use of to find similar questions within the predetermined categories. The experimental results show that our approach leads to a better performance than other approaches.

Keywords: Community question answering, Question search, Categorization, Dependency syntactic tree.

1 Introduction

In the last few years, many CQA systems have been launched, including Yahoo! Answers, BuyAns, Live QnA. Since their inception, CQA sites have rapidly gained popularity. Hundreds of millions of answers have already been posted for tens of millions of questions in Yahoo! Answers. When a user chooses to ask a new question in a CQA service, the user would have to wait for a long time before getting answers. If the CQA service could automatically search and display relevant pre-existing questions and their answers, it will reduce the waiting time and improve the user satisfaction. Hence, it's important the search service offers relevant results efficiently.

Instead of inputting just keywords or so, users form questions using natural language. Many previous approaches introduce natural language models to find similar questions, such as translation model [1], MDL tree [3], syntactic tree [4] and so on [5]. They focus on the text contents, weakening the structure information of forums. Some researchers have utilized some features of CQA sites. Sun et al. [6] did the question retrieval based on user ratings. The category

* Corresponding author.

information of questions is proven to be positive information for question retrieval [7]. It can prune the search space and improve the efficiency and the effectiveness. However, the utilization of features is not enough.

In this paper, we propose a retrieval model that utilizes more features of CQA sites, including question, description, answer, category and users' posted questions. The model can be divided into two parts. First, the category classification of a query is predetermined according to the asker's posted questions. Second, we make use of "dependency syntactic tree" to find similar questions within the predetermined categories.

2 Related Work

Many previous approaches introduce natural language models to find similar questions. Xue et al. [1] proposed a retrieval model that combined a translation-based language model for the question part with a query likelihood approach for the answer part. Joen et al. [2] discussed methods for question retrieval that were based on using the similarity between answers in the archive to estimate probabilities for a translation-based retrieval model. Duan et al. [3] proposed to use the MDL-based tree cut model for identifying question topic and question focus automatically. Wang et al. [4] tackled the similar question matching problem based on syntactic tree structure. The model did not rely on training.

Some researchers have solved some problems in the CQA by exploring the features of CQA sites. Sun et al. [6] utilized the question star feature to recommend new question to the askers who posted similar questions. Liu et al. [9] developed a variety of content, structure, and community-focused features for user satisfaction prediction. Jeon et al. [10] used textual and non-textual features that were commonly recorded by web services to improve the search quality.

Closest to our work, Cao et al. presented a new approach [8] to exploit category information of questions for improving the performance of question retrieval. In the paper [7], they used a category language model to smooth a question language model, and integrated the classification scores returned by a classifier built with historical question data into language models. Our work differs from it on the classification and the retrieval model. Our classification is based on the users' posted questions, not only the category. The similar questions are retrieved with the "keywords" extracted from "dependency syntactic tree".

3 Category Based Question Classification

3.1 Category Probability Table Construction

Given a new question q , the probability of the question belonging to a category c for the asker u is defined as $p(c|q, u)$. A question can be represented by a set of words. Then the computation for $p(c|q, u)$ depends on the computation for $p(c|w, u)$. After tokenization, stop words filtering and stemming, $p(c|w, u)$ is estimated as follows:

$$p(c|w, u) = freq(w, c|u) \times con(w, c|u), \quad (1)$$

where $freq(w, c|u)$ represents the frequency of the word w in the category c with the user u , and $con(w, c|u)$ is the correlation of the word w to the category c for the user u . They are estimated as follows:

$$freq(w, c|u) = \frac{n(q|c, w, u)}{n(q|c, u)}, \quad (2)$$

where $n(q|c, w, u)$ denotes the number of the user u 's posted questions which contain the word w in the category c , and $n(q|c, u)$ is the number of the user u 's posted questions in the category c .

$$con(w, c|u) = \frac{1}{n(c|w, u)}, \quad (3)$$

where $n(c|w, u)$ denotes the number of categories where the word w occurs for the user u . The lower the value, the more correlated between the word and the category.

3.2 Question Classification

We apply the max algorithm to compute the top- k ranked results for a new question query. The benefit of the max algorithm is that $p(c|q, u)$ depends on the frequent words in the specific category, ignoring the effect of noise words. Given a query q containing l words and the asker u , the probability of the question q belonging to the category c can be obtained by:

$$p(c|q, u) = \max_{w \in q} p(c|w, u), \quad (4)$$

where $p(c|w, u)$ is the probability of the word w belonging to the category c with the user u

4 Similar Questions Retrieval

Dependency syntactic tree analysis is a research focus in the field of Natural Language Processing. A new query is represented by a "dependency syntactic tree" with the Minipar [11]. In the tree, the connection between the nodes reflects the dependency between them. The category attribute of node reflects POS (part of speech).

Definition 1 (Keywords). *keywords consists of two parts. One is single noun "singlekey" except stop words. The other is double words "bikeys" (bikeys₁ bikeys₂) from "dependency syntactic tree". The word bikeys₁ has dependency relationship with the word bikeys₂. The POS of them is limited to noun, verb, adjective and adverb. In addition, the keywords are processed with stemming.*

The candidate questions are obtained by searching the questions that have at least one the same keyword of the query q in the field of question/description/

answer within the top- k similar categories. The method utilizes all the information of a question. Though a question has no literal similarity with the query, if the answer field has keywords, it will be a candidate question.

The candidate questions are ranked by taking the addition across all the weight of the keywords in the question:

$$score = \sum_{k \in keywords} \log \frac{n(q|c)}{n(q|k, c)}, \quad (5)$$

where $n(q|c)$ denotes the number of questions in the category c , and $n(q|w, c)$ is the number of questions which contain the keyword k in the category c .

5 Experiments

5.1 Experimental Setup

Dataset. Our data was based on a snapshot of Yahoo! Answers, containing 4,773,563 questions. It was crawled in the late 2012. There are 26 categories at the first level and 1263 categories at the leaf level. Each question belongs to a unique leaf-level category. We randomly selected 100 users whose activities are public in each top-level category, 2600 users in total. Then we collected the questions that have been posted by the 100 users as the users' posted questions set. 300 questions are randomly selected from the questions set as the test data set, the rest of questions set is as the training data for question classification. We used the Lucene to preprocess the text, including tokenization, stop word filtering and stemming.

We remove the stop words. For each model, the top 20 retrieval results are kept. We put all the results from different models for one query question together for annotation. Thus annotators do not know which results are from which model. Annotators are asked to label each returned question with "relevant" or "irrelevant". Two annotators are involved in the annotation process. If conflicts happen, a third person will make judgment for the final result. We eliminate the query questions that do not have relevant questions. Finally we get 246 queries that have relevant questions.

Evaluation Metrics. We evaluate the performance of our approaches using Mean Average Precision (MAP), Mean reciprocal rank (MRR) and Precision@n.

MAP rewards approaches returning relevant questions earlier, and also emphasizes the rank in returned lists. MRR gives us an idea of how far down we must look in the ranked list in order to find a relevant question. Precision@n is the fraction of the Top- n questions retrieved that are relevant. The evaluation was based on the top 20 results from all approaches.

Methods Compared. We compare with the Translation Model (TR) that have been used for question retrieval in the paper [2], category based retrieval [7] (CR). We also report the results of the Vector Space Model(VSM), the Okapi Model and the Language Model (LM).

5.2 Experimental Results

Each question belongs to a unique leaf-level category by the classification algorithm. Question search can benefit from classification if the correct category is contained in the Top- n returned categories. In order to see if the correct category is contained in the Top- n returned categories, we compute the percentage of test query questions whose correct categories are contained in the Top- n categories returned by the classification algorithm. The paper [7] called the percentage "Success@n." It is shown in Figure 1(a). We can see that the accuracy of question classification in the Top-10 categories returned by the classification algorithm of our method "C&MR" reaches 80%. There are 1263 leaf-level categories. It greatly prunes the search space.

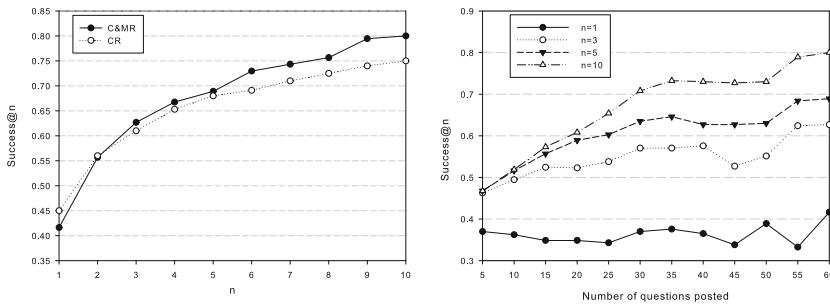


Fig. 1. (a) Success@n: the percentage of question whose correct categories are contained in Top-n returned categories by the classifier; (b) Success@n for question classification with varying number of users' posted questions, $n=1,3,5,10$

Figure 1(b) shows the performance of question classification with varying number of history data. The abscissa shows the number of users' posted questions for training. Then we use users' latest 10 questions to test. History questions have nearly no effect in the situation when the correct category is at the first rank. However, it becomes helpful when the correct category in the top-3/5/10 categories returned by the classification algorithm, and the "success@n" increases with the more posted questions as the training set. The percentage may reach 80% when the number of history data is 60.

As shown in Table 1, our approach "C&MR" significantly outperforms other approaches, and performs slightly worse than CR on the metric of P@n. The relevant questions returned by C&MR have high rank. The number of returned relevant questions is a little less than that of CR. This is because the keywords extracted from "dependency syntactic tree" search the match in the three features: question, description and answer. The match in the three features has the same weight. The relevant questions with more keywords match will get high rank score. It rewards the highly similar questions. The relevant question which is only similar in the question feature will get low rank score.

Table 1. Performance of Different Approaches Measured by Different Metrics

	VSM	Okapi	TR	LM	CR	C&MR
MAP	0.2457	0.3423	0.4053	0.3879	0.4704	0.6437
MRR	0.4453	0.5406	0.6084	0.5945	0.6649	0.7414
p@5	0.2222	0.2857	0.3168	0.3040	0.3476	0.3222
p@10	0.1683	0.2242	0.2438	0.2310	0.2623	0.2555

6 Conclusions

In this paper, we propose a retrieval model that utilizes more features of CQA sites, including question, description, answer, category and users' posted questions. The model is divided into question classification and question retrieval. First, the category classification of a query are predetermined according to the asker's posted questions. Second, "dependency syntactic tree" is made use of to find similar questions within the predetermined categories. We have demonstrated the effectiveness of our approach in large-scale experiments

Acknowledgments. The work described in this paper was fully supported by the National Natural Science Foundation of China under Grant No. 61170184.

References

1. Xue, X., Jeon, J., Croft, W.B.: Retrieval models for question and answer archives. In: Proc. of SIGIR, pp. 475–482 (2008)
2. Jeon, J., Croft, W.B., Lee, J.H.: Finding similar questions in large question and answer archives. In: Proc. of ACM CIKM, pp. 84–90 (2005)
3. Duan, H., Cao, Y., Lin, C., Yu, Y.: Searching questions by identifying question topic and question focus. In: Proc. of ACL, pp. 156–164 (2008)
4. Wang, K., Ming, Z., Chua, T.: A syntactic tree matching approach to finding similar questions in community-based QA services. In: Proc. of SIGIR, pp. 187–194 (2009)
5. Song, Y., Lin, C., Cao, Y., Rim, H.: Question Utility: A Novel Static Ranking of Question Search. In: Proc. of AAAI, pp. 1231–1236 (2008)
6. Sun, K., Cao, Y., Song, X., Song, Y., Wang, X., Lin, C.: Learning to recommend questions based on user ratings. In: Proc. of ACM CIKM, pp. 751–758 (2009)
7. Cao, X., Cong, G., Cui, B., Jensen, C.S., Zhang, C.: The use of categorization information in language models for question retrieval. In: Proc. of ACM CIKM, pp. 265–274 (2009)
8. Cao, X., Cong, G., Cui, B., Jensen, C.S.: A generalized framework of exploring category information for question retrieval in community question answer archives. In: Proc. of WWW, pp. 201–210 (2010)
9. Liu, Y., Bian, J., Agichtein, E.: Predicting information seeker satisfaction in community question answering. In: Proc. of ACM SIGIR, pp. 483–490 (2008)
10. Blooma, M.J., Chua, A.Y., Goh, D.H.: A predictive framework for retrieving the best answer. In: Proc. of ACM SAC, pp. 1107–1111 (2008)
11. Lin, D.: Principle-Based Parsing Without OverGeneration. In: Proc. of ACL, pp. 112–120 (1993)

Ranking Web Pages by Associating Keywords with Locations

Peiquan Jin, Xiaoxiang Zhang, Qingqing Zhang, Sheng Lin, and Lihua Yue

University of Science and Technology of China, 230027, Hefei, China
jpcq@ustc.edu.cn

Abstract. Many Web queries contain both textual keywords and location words. When answering such queries, the association between the textual keywords and locations in a Web page should be taken into account. In this paper, we present a new ranking algorithm for location-related Web search, which is called MapRank. Its main idea is to extract the associations between keywords and locations in Web pages and further use them to improve ranking effectiveness. We first determine map each keyword with specific locations and form a set of \langle keyword, location \rangle pairs. Then, we compute the location-constrained score for each keyword and combine it into the ranking procedure. We conduct comparison experiments on a real dataset and use the metrics including MAP and NDCG to measure the performance of MapRank. The results show that MapRank is superior to previous methods with respect to different symbolic-location-related queries.

Keywords: ranking algorithm, symbolic location, Web search, association.

1 Introduction

Ranking algorithms, e.g., the *Pagerank* algorithm, have been one of the major technologies in search engines. Unfortunately, traditional ranking algorithms are based on link analysis and textual relevance, and are hard to satisfy different querying needs. Besides, the textual-relevance-based ranking approach does not consider the relationship between textual keywords and location names in a query. On the other hand, many Web pages are associated with certain locations, e.g., news report, retailer promotion and so on. The study in the literature [1] reported that among 2,500 queries, 18.6% of them contained a geographic predicates and 14.8% of them included a location name. Therefore, how to extract locations for Web pages and use them in Web search has been a hot and critical issue in current research on Web search [2-4].

In this paper, we present a new location-aware ranking algorithm for Web search, which is called *MapRank*. MapRank aims to improve the ranking performance for spatial textual Web queries that contain both textual keywords and location words. The algorithm considers both textual and location relevance between Web pages and querying terms when returning the results, and can improve the effectiveness of Web search engines. The contributions of the paper can be summarized as follows:

- (1) We propose a new ranking algorithm named *MapRank* for spatial textual Web queries. MapRank is implemented using a two-staged strategy, namely an offline

stage extracting and building $\langle keyword, location, score \rangle$ pairs for Web pages and an online stage computing the final ranking score. The new algorithm considers both textual and location relevance in the ranking process, and also takes into account the relationship between keywords and locations in a Web page.

(2) We conduct comparison experiments on various real datasets crawled from New York Time, to measure the performance of the MapRank algorithm. The experimental results show that the proposed MapRank algorithm has the best performance with respect to different spatial textual queries.

2 Related Work

Spatial textual queries are usually represented as a triple $\langle what, relation, where \rangle$. However, as the “In” relation is the most appropriate one for Web search, spatial textual queries in Web search engines can be simplified as $\langle what, where \rangle$. Location-related ranking algorithms are specially designed to cope with spatial textual queries in Web search engines. Basically, a location-related ranking algorithm has to consider both textual relevance and location relevance between query and Web pages. The challenging issues are to determine location relevance and combine textual and location relevance during the ranking process.

There are two major methods to compute the location relevance. One of them is to utilize the relation words in queries [5-7]. For example, many spatial queries contain some spatial relation words such as “inside”, “overlap”, and “nearby”. The other type does not use the explicit spatial relation words in queries, but determines spatial relations based on the geographic attributes of the spatial objects in queries [8, 9]. For instance, the spatial object “Paris” in the query can be mapped into a determined geographic extent.

Answering spatial textual queries need to consider both textual and location relevance. The native way is to combine them using a linear-weighted method. Some other works do not use the combination of textual relevance and location relevance. Li et al. [10] introduced a topic model to determine the topic of Web queries as well as Web pages. As a topic usually has a distribution among geographic extent, they proposed to utilize a Gauss formula to simulate the geographic distribution of a topic. The geographic distribution model of topics is then taken to determine the ranking scores of Web pages. Some researchers proposed ranking mechanisms which combines various metrics in textual relevance computation and location relevance determination. For example, Martins et al. [11] proposed an SVM-based optimized MAP method, called SVMmap, and Cai et al. [12] proposed the GeoVSM, which is a geographic optimized VSM model.

More recent works on spatial textual query processing are conducted by Gao Cong et al. [2, 3], and a spatial textual search engine called SWORS [2] is designed. The locations in SWORS are with the similar semantics in traditional geographical information systems. Therefore, spatial queries involving geographical relations such as “nearby” and “close to” have to be resolved by some new indexing structures, e.g., IR-Tree [3]. Regarding the ranking techniques, SWORS considered both textual and spatial relevance, which was similar with the previous solutions.

3 The MapRank Algorithm

3.1 The Basic Idea

MapRank is a location-aware ranking algorithm for spatial textual Web search. It considers both textual relevance and location relevance of Web page. The basic idea of MapRank can be described as follows:

(1) MapRank considers the association of keywords and locations when computing the scores of Web pages. In particular, we map each keyword in a Web page with a specific focused location and then calculate the location-constrained score of each keyword. As a result, we construct a set of $\langle \text{keyword}, \text{location}, \text{score} \rangle$ pairs for each Web page, in which the location represents the most relevant focused location of the given keyword.

(2) We use a two-staged design to implement the MapRank algorithm. The first offline stage is to construct the $\langle \text{keyword}, \text{location}, \text{score} \rangle$ pairs for each Web page. The second online stage is to compute the final ranking scores for all the Web pages. In the second stage, we combine two factors, namely the relevance between the querying location and the focused locations in Web page, and location-constrained keyword score, to achieve a tradeoff between location relevance and textual relevance.

The main difference between MapRank and other existing ranking algorithms is that it combines the focused locations of Web page into the ranking algorithm. Furthermore, the $\langle \text{keyword}, \text{location}, \text{score} \rangle$ mapping policy also introduces a reasonable solution to integrate textual and location relevance into the ranking algorithm.

3.2 Constructing $\langle \text{keyword}, \text{location}, \text{score} \rangle$ Pairs

The focused location refers to the most appropriate location associated with a Web page. The extraction of focused locations is performed by the algorithm discussed in our previous work [13]. It returns a set of focused locations for each Web page.

Generally, a spatial textual query contains several keywords and one location word. Moreover, the keywords and the location word in most spatial textual queries usually imply some relationships which represent users' indeed searching needs. For example, a query "Massachusetts population statistics" actually means that users want to find the population statistics of the state "Massachusetts". Here, the text keywords "population statistics" and the location word "Massachusetts" in the query have an intrinsic relationship. In general, the relationship between a keyword and a location can be represented as a pair $\langle \text{keyword}, \text{location} \rangle$, which indicates that the given keyword is mostly related with the location.

Current search engines deals with the text keywords and location words individually and ignores the relationship between the keywords and locations. Our MapRank algorithm is designed to present a better solution for this problem. We will consider the relationship between keywords and locations when performing the ranking task. This idea is motivated by the temporal textual ranking approach proposed in [14]. Basically, we first find the most relevant focused location for each keyword, and construct $\langle \text{keyword}, \text{location} \rangle$ pairs. After that, we compute the location-constrained ranking score of each keyword, which finally forms the list of $\langle \text{keyword}, \text{location}, \text{score} \rangle$.

We use three constant values to measure the scores for $\langle \text{keyword}, \text{location} \rangle$ pairs, namely TITLE_SCORE, SENT_SCORE, and PARA_SCORE. The TITLE_SCORE

is used to represent the score of $\langle \text{keyword}, \text{location} \rangle$ when the keyword and associated location word are both contained in the title of the Web page. The SENT_SCORE is used when the keyword and associated location word are both contained in the same sentence, but not in the title. The PARA_SCORE is used when the keyword and associated location word are both contained in the same paragraph, but not in the same sentence. Generally, we have the following assumption: TITLE_SCORE > SENT_SCORE > PARA_SCORE.

3.3 Computing the Ranking Scores of Web Pages

When users post a query to the search engine, the final ranking scores of Web pages are computed dynamically. In this paper, the final ranking score of a Web page is calculated based on two factors, namely the relevance between the querying location and the focused locations in Web page, and location-constrained keyword score.

Given a Web page D , a set of querying keywords, say $\langle w_1, w_2, \dots, w_n \rangle$, and a querying location g , the computation of the final ranking score is based on the following algorithm (as shown in Fig.1).

Algorithm *Location_Ranking*(D, Q)

Input: (1) a Web page D , which has the list of $\langle \text{keyword}, \text{location}, \text{score} \rangle$ pairs $K = \{ \langle k_1, l_1, sc_1 \rangle, \langle k_2, l_2, sc_2 \rangle, \dots, \langle k_m, l_m, sc_m \rangle \}$.
(2) a query Q including a keywords set $W = \langle w_1, w_2, \dots, w_n \rangle$ and a location g

Output: GS , the ranking score of D according to Q

Preliminary: n is the count of focused locations.

/ Removing the keywords unrelated with Q from K */*

```

1: for each  $\langle k, l, sc \rangle \in K$  do
2:   if  $k \notin W$  then
3:      $K = K - \langle k, l, sc \rangle$ ; // remove unrelated keywords
/* Computing ranking score */
4: for each  $\langle k, l, sc \rangle \in K$  do
5:    $GS = GS + \frac{\text{common}(l, g)}{\max(l, g)} \cdot sc$ 
6: return  $GS$ ;

```

Fig. 1. Computing the ranking score for Web page

In Fig.1, we emphasize the location relevance between the user query and Web pages. This is done by introducing location relevance into the computing procedure of ranking scores, as defined by the formula 3.1.

$$\text{location relevance} = \frac{\text{common}(l, g)}{\max(l, g)} \quad (3.1)$$

Given two locations l and g , $\text{common}(l, g)$ in Formula 3.1 is defined as the length of the common prefix of l and g in the location tree generated from Gazetteer, and $\max(l, g)$ refers to the maximum length of l and g . For example, suppose that

l = “USA/Massachusetts/Peak Stone”, g = “USA/Arizona”, $\text{common}(l, g)$ is 1 and $\text{max}(l, g)$ is 3.

4 Experimental Results

To evaluate the performance of MapRank, we conduct an experiment on a dataset crawled from New York Time, which contains 311,187 Web pages ranging from 2006 to 2011. For every Web page in the dataset, we cut the page into words using HTMLParser (<http://htmlparser.sourceforge.net/>) and use using the stop-words database provided by SMART (<http://www.lextek.com/manuals/onix/stopwords2.html>) to filter stop words. Then we stem the keywords using the Porter Stemmer tool (<http://tartarus.org/~martin/PorterStemmer/>). After that, we extract the focused locations for each Web page and construct $\langle \text{keyword}, \text{location}, \text{score} \rangle$ pairs. The extracted keywords are maintained in Lucene 3.5, which will be used when executing comparison algorithms.

In the experiments, we run 16 spatial textual queries and use two metrics to measure the performance of each algorithm, i.e., MAP and NDCG. We implement the algorithms using Java under the developing environment ObjectWebLomboz. The test machine has an Intel Dual Core Processor, 2GB of main memory, and is running Windows XP Professional.

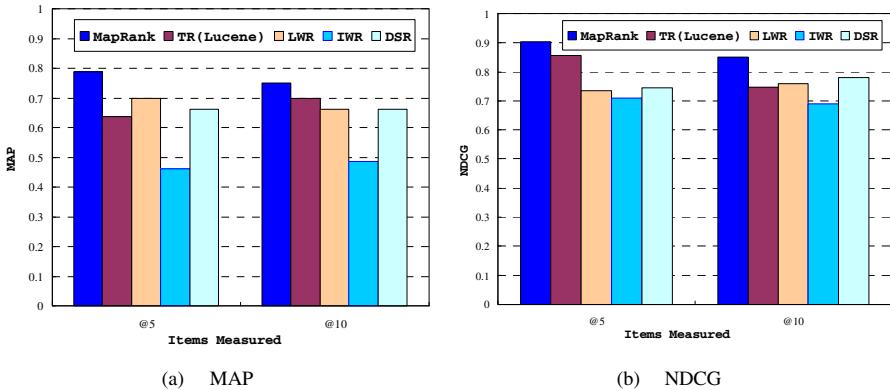


Fig. 2. MapRank vs. other four comparison algorithms

Figure 2(a) shows the MAP@5 and MAP@10 scores of MapRank and other four competitor algorithms, namely Textual Ranking (TR), Linear Weighted Ranking (LWR), Improved Weighted Ranking (IWR), and DS Ranking (DSR). The LWR approach uses the linear weighted sum of textual relevance and location relevance as the ranking score of Web page. The IWR approach is similar with the linear weighted ranking one, except that it uses an improved way to determine the parameter ω . In particular, it uses the method in the literature [15] and determines the ω value on the basis of the description of textual words and location words. The DSR approach is based on the Dempster-Shafer (DS) theory of evidence, and uses different types of evidences to determine the possibility of an event [15]. The textual relevance and location relevance can be regarded as two individual evidences for spatial textual

ranking. In our experiment, we use the weight of importance for each textual word and location word to determine the uncertainty of those two evidences, and rank Web pages according to the uncertainty. Figure 2(b) shows the different NDCG scores. As shown in Fig.2, MapRank gets the best MAP and NDCG scores in all cases.

5 Conclusion

In this paper, we introduce the MapRank algorithm which is based on the association between the focused locations of Web page and keywords. It presents an appropriate tradeoff between textual relevance and location relevance. The experimental results show that the MapRank algorithm has better performance for spatial textual queries than its competitors. Next we will integrate our algorithm with temporal information in Web pages.

Acknowledgement. This paper is supported by the National Science Foundation of China (No. 60776801 and No. 71273010), the National Science Foundation of Anhui Province (no. 1208085MG117), and the USTC Youth Innovation Foundation.

References

1. Sanderson, M., Kohler, J.: Analyzing geographic queries. In: Proc. of GIR (2004)
2. Cao, X., Cong, G., Jensen, C.S., et al.: SWORS: A System for the Efficient Retrieval of Relevant Spatial Web Objects. *PVLDB* 5(12), 1914–1917 (2012)
3. Cong, G., et al.: Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects. In: Proc. of VLDB (2009)
4. Lu, J., Lu, Y., Cong, G.: Reverse Spatial and Textual K Nearest Neighbor Search. In: Proc. of SIGMOD, pp. 349–360 (2011)
5. Zhou, Y., Xie, X., Wang, C., et al.: Hybrid Index Structures for Location-based Web Search. In: Proc. of CIKM, pp. 155–162. ACM, New York (2005)
6. Martin, B., Silva, M., et al.: Indexing and Ranking in Geo-IR Systems. In: GIR 2005 (2005)
7. Andrade, L., et al.: Relevance ranking for geographic information retrieval. In: GIR 2006 (2006)
8. Jones, C.B., Alani, H., Tudhope, D.: Geographical Information Retrieval with Ontologies of Place. In: Montello, D.R. (ed.) COSIT 2001. LNCS, vol. 2205, pp. 322–335. Springer, Heidelberg (2001)
9. Larson, R.: Ranking approaches for GIR. *SIGSPATIAL Special* 3(2) (2011)
10. Li, H., Li, Z., Lee, W.-C., et al.: A Probabilistic Topic-Based Ranking Framework for location-sensitive domain information retrieval. In: Proc. of SIGIR, pp. 331–338 (2009)
11. Martins, B., Calado, P.: Learning to Rank for Geographic Information Retrieval. In: Proc. of GIR (2010)
12. Cai, G.: GeoVSM: An Integrated Retrieval Model for Geographical Information. In: Proc. of GIS, pp. 65–79 (2002)
13. Zhang, Q., Jin, P., Lin, S., Yue, L.: Extracting Focused Locations for Web Pages. In: Wang, L., Jiang, J., Lu, J., Hong, L., Liu, B. (eds.) WAIM 2011 Workshops. LNCS, vol. 7142, pp. 76–89. Springer, Heidelberg (2012)
14. Jin, P., Li, X., Chen, H., Yue, L.: CT-Rank: A Time-aware Ranking Algorithm for Web Search. *Journal of Convergence Information Technology* 5(6), 99–111 (2010)
15. Yu, B., Cai, G.: A Query-Aware Document Ranking Method for Geographic Information Retrieval. In: Proc. of GIR, pp. 49–54. ACM, New York (2007)

Behavioral Consistency Measurement and Analysis of WS-BPEL Processes

Xuewei Zhang^{1,3}, Wei Song^{2,3}, Jianchun Xing¹, Qiliang Yang¹,
Hongda Wang¹, and Wenjia Zhang²

¹ PLA University of Science and Technology, Nanjing, China

² Nanjing University of Science and Technology, Nanjing, China

³ State Key Laboratory of Software Engineering, Wuhan University, Wuhan, China

wsong@njjust.edu.cn

Abstract. With the development of services and cloud computing, service-based business processes (e.g., WS-BPEL processes) are paid more attention by practitioners. Business parties usually keep their BPEL processes in a process repository. In order to facilitate the retrieval, maintenance, and reuse of BPEL processes in the repository, we need an appropriate measurement criterion to analyze the behavioral consistency between BPEL processes. In this paper, we propose a novel measurement criterion and corresponding analysis approach to determine the behavioral consistency between two BPEL processes quantitatively. Our measurement and approach are based on BPEL program dependence graphs (BPDGs). We have faithfully implemented our approach in a prototype tool which is used to analyze the behavioral consistency of BPEL processes.

Keywords: BPEL Processes, Behavioral Consistency, Process Alignment, BPEL Program Dependence Graph.

1 Introduction

In SOA (Service-Oriented Architectures), Web Services Business Process Execution Language (WS-BPEL [1] or BPEL) is the de facto standard for service-based business processes. With BPEL processes enriching and expanding constantly, many Business parties build the BPEL process repositories. In order to retrieve, maintain and reuse these BPEL processes better, an appropriate criterion is indispensable to determine the consistency between BPEL processes. For example, when one wants to insert a new BPEL process into a process repository, he first ought to make sure that the same BPEL process is absent from the repository. Then, he further needs to determine which BPEL processes are similar to the newly-added one. In such situations, we need to study the consistency degrees of different BPEL processes.

Different criteria can be used to formulate the consistency between two BPEL processes. For instance, we can study this issue from the interface perspective and define the consistency degree based on the match degree of interfaces. However, this criterion is not appropriate because the behaviors of two BPEL processes may quite different even if they share the same set of interfaces. In this paper, we study the consistency between two BPEL processes from the behavioral perspective.

One can utilize different notions of equivalence in the linear time-branching time spectrum [2] to define the behavioral consistency between two process models. For example, *trace equivalence* is commonly seen as the lower bound of this spectrum, but it is still too strict to determine the behavioral consistency between process models [3] due to the following shortcomings. First, it is more sensitive to projections. If the additional start and end branches are introduced in a process model, the consistency degree of process models before and after change are greatly affected. Second, it only provides a “true/false” answer, which prevents it from finding different but similar process models. To this end, Weidlich et al. propose in [3] an approach based on *behavioral profile* which captures the essential behavioral constraints between activities in process models. Informally, two process models are consistent if their behavioral constraints are consistent for every aligned activity pairs. This approach is less sensitive to projections than the approach based on trace equivalence. Furthermore, it can return a consistency degree ranging from 0 to 1. Obviously, the approach that is applied in the field of process models is very excellent. As we know, BPEL process is executable and it contains much specific information, such as partnerLinks, the variables of input and output. Unfortunately, the important information has been abstracted away in process models, thus, behavioral profiles are inappropriate to determine the consistency degrees of BPEL processes.

To address this problem, we will propose in this paper a behavioral consistency measurement of BPEL processes. In the software engineering community, consistency refers to a “degree of uniformity, standardization, and freedom from contradictions” [4]. Similarly, we define the consistency between BPEL processes as a “degree of invariance of the activity relations and freedom from contradiction”. The activity relations considered here include exclusiveness and three kinds of “happens-before” activity dependences (i.e., control dependence, data dependence, and asyn-invocation dependence). These activity relations can be captured into a *BPEL program dependence graph* (BPDG) [5], [6]. Based on BPDGs, we present our consistency measurement and the corresponding analysis approach to determine the consistency degree of two BPEL processes.

The contribution of our work is threefold. First, we propose a novel approach to quantitatively determine the behavioral consistency between two BPEL processes based on their BPDGs, rather than as the conventional approaches only provide a qualitative conclusion of consistence or inconsistency. Second, we implement a prototype tool to calculate the consistency degree of two BPEL processes. Third, we apply our approach by analyzing the consistency between some real-life BPEL processes.

The remainder of the paper is structured as follows. Some preliminaries are introduced in Section 2. In Section 3, we propose our measurement and the analysis approach to partnerLinks the consistency degree of two BPEL processes. In Section 4, we discuss some implementation issues for our prototype tool and report our findings of applying our approach in practice. We discuss the limitations of our approach and review related work in Section 5 and 6, respectively. Section 7 concludes the paper.

2 Preliminaries

In this paper, we leverage behavioral constraints to quantify the consistency between BPEL processes. To facilitate users to understand our approach better, we review some preliminaries in this section. We show the input and output of each activity of a

BPEL process that is associated with the corresponding node (activity) in the BPEL Control Flow Graph (BCFG) [7]. Thus, not only control dependence but also data dependence between activities can be analyzed. To make this paper self-contained, we first review the concepts of control and data dependences [8].

Activity A_j is control dependent on its prior activity A_i , iff A_i determines whether A_j is executed in BPEL process. Generally speaking, A_i denotes some conditions activities, such as <if>, <while>. If activity A_j is not controlled by any activity, we assume that A_j is control dependent on *Entry* node. For example, nodes C_1 , C_2 , D_1 and D_2 are control dependent on node B , and *Entry* node controls nodes A , B and E in Fig.1.

Data dependences can be classified into three categories in BPEL process: true-data dependence, anti-data dependence and output-data dependence [8]. The anti-data and output-data dependences can be avoided through the variables renaming, so only the true-data dependence is considered in the paper. We can discuss Fig.1 again. Since node B uses the output of node A , node B is true-data dependent on node A . Analogously, nodes E , C_1 and C_2 are respectively true-data dependent on nodes D_1 or D_2 .

Apart from control and data dependencies, it defines a novel dependence, namely, asyn-invocation dependence [5]. This dependence is caused by the asynchronous communicating mechanism between BPEL processes. In Fig.1, we can see that activities C_1 and C_2 are two asynchronous <invoke> activities, and the responses of activities C_1 and C_2 are respectively received by activities D_1 and D_2 .

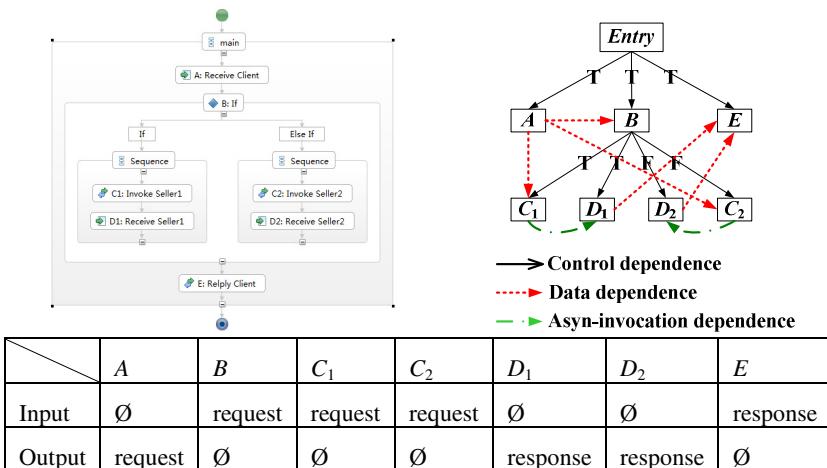


Fig. 1. TCFG and BPDG

BPDG represents a BPEL process as a directed graph where these nodes denote basic or structure activities except for *Entry* node and the edge indicates some dependences between two nodes. We can present the following definition of BPDG.

Definition 1 (BPEL Program Dependence Graph, BPDG) [5]. A BPEL program dependence graph is a directed graph $\langle N, E \rangle$, where

1. N is a set of nodes. There is one *Entry* node in N while the other nodes denote the activity statements and predicate expressions.

2. $E \subseteq N \times N$ is a set of directed edges, and an edge $\langle X, Y \rangle \in E$ directed from X to Y denotes the control, data, or asyn-invocation dependence between the two activities or predicate expressions represented by X and Y .

In this paper, we capture these three kinds of dependences in BPDG to analyze the consistency of BPEL processes. The set of all dependences for BPEL process may be viewed as inducing a partial ordering on the activities and predicates that must be followed to preserve the semantics of the original process. BPDG makes the dependences explicit between activities. We can realize that as only one branch can be executed in the selective structures of BPEL process, there is exclusiveness relation between activities on these different branches. In order to accurately describe BPEL process, each control dependence edge can be labeled “True” (“T”) or “False” (“F”). Additionally, the data dependence edges should be attached to correlated variables, but the labels are omitted in order to conveniently denote them.

Actually, it not only limits to the three ones for the dependence relations of BPDG, that is, extendibility. We are also able to put forward other appropriate dependences. For example, when only considering some part or variable of a BPEL process, we can get the dependences about the part or variable based on user-defined dependence. It helps to simplify and understand BPEL process.

3 Consistency Measurement and Analysis

In this section, we present our approach which measures the consistency between BPEL processes based on behavioral constraints. First, the consistency measurement of BPEL processes is defined in Section 3.1. Then, based on four instances, Section 3.2 analyzes the consistency degree of two BPEL processes and illustrates the advantages of our approach.

3.1 Consistency Measurement of BPEL Processes

In the paper, we assume that the name of each activity is unique, and they are corresponding only if some attributes of activities from different BPEL processes are the same. The attributes contain name, types, partnerLink and portType. The alignment problem addresses the question whether two BPEL processes are consistent given a set of correspondences between their activities. The alignment can be further classified into two categories: vertical alignment and horizontal alignment [9]. Horizontal alignment can generate a 1:1 correspondence between activities in two BPEL processes, and there are 1:2 correspondences between activities for vertical alignment. However, the consistency degree between aligned BPEL processes is independent on the distinction of vertical and horizontal alignment.

In practice, there are some differences for an alignment in BPEL processes and process models, and a different part is shown in Fig.2. Due to asyn-invocation dependence between activities B_1 and B_2 in model (b), activity B from model (a) are 1: 2 correspondences to both, namely, vertical alignment. Activity B is horizontal alignment to activities B_1 in process models, but activity B_2 is discarded as there are no correspondences in model (a). Actually, there might be n: m correspondences between activities of the aligned process models in the different abstract level. The situation does not exist in BPEL processes. If an activity is discarded in BPEL process, we consider that they are preserved for data dependences between it and other activities.

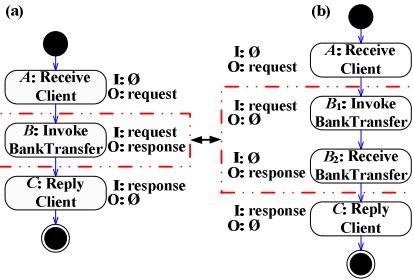


Fig. 2. Activity A is 1:2 correspondences to activities A_1 and A_2

In general, our notion of BPEL processes consistency based on BPDGs is grounded on the preservation of behavioral relations for the corresponding activities. Thus, the problem of BPEL process consistency is transformed into the problem of activity constraint consistency. In order to specify the degree of behavior that is preserved by an alignment, we define two sets of consistently aligned activity pairs, one for each of the aligned BPEL processes. These sets contain all pairs of activities that are aligned by a correspondence relation, and their relations with BPDGs preserving. Then, we define the consistency degree of BPEL processes following:

Definition 2 (Degree of Consistency of Alignment of BPEL Processes). *For the threaded control flow graphs of two aligned BPEL processes, the degree of consistency based on BPDGs of \sim is defined as*

$$P_B^\sim = \frac{|C_1^\sim| + |C_2^\sim|}{|N_1^\sim \times N_1^\sim| + |N_2^\sim \times N_2^\sim|}.$$

The corresponding relation $\sim \subseteq (N \times N)$ aligns both processes by relating corresponding activity pairs to each other such that $\sim \neq \Phi$. $N_1^\sim \subseteq N_1$ and $N_2^\sim \subseteq N_2$ denote the sets of the aligned activities, and C_1^\sim and C_2^\sim denote the sets of the consistent activity pairs.

3.2 Consistency Analysis

We have the following three steps to analyze the consistency degree of two BPEL processes (cf. Fig. 3). First, two BPEL processes are transformed into two BCFGs in which each node is associated with some relevant information, such as the name of activity, the variables of input and output, partnerLinks, etc. Second, BPDGs of the two BPEL processes are derived by control, data, and asyn-invocation dependences analysis based on BCFGs. Note that when there are `<if>`, `<switch>`, `<pick>` activities in a BPEL process, since only a branch might be executed for this kind of structured activities, there is exclusiveness relation between any two activities on different branches. Third, we calculate the consistency degree of the two BPEL processes based on the obtained BPDGs. In fact, control dependence and data dependence have the different influence for the consistency degree of BPEL processes. We can realize that once control dependence of the corresponding activity of a BPEL process is changed, they are inconsistent for all corresponding activity pairs that contain the activity of the two BPEL processes, but data dependence may only affect the corresponding activity pair.

We find that the notion of trace equivalence is very strict, for example, if the executing order of activities is changed slightly, it might weaken the consistency degree

heavily based on trace equivalence, even result in the degree being *zero*. Behavioral profile defines three behavioral constraints between activities in process model. The approach works wonderfully for checking the consistency degree in process models, but it is not appropriate to BPEL processes because they contain much data information. As BPDG captures the essential dependence relations between activities in BPEL process, it describes the behavior constraints of BPEL process roundly. The approach is applicative to measure the consistency degree of BPEL processes.

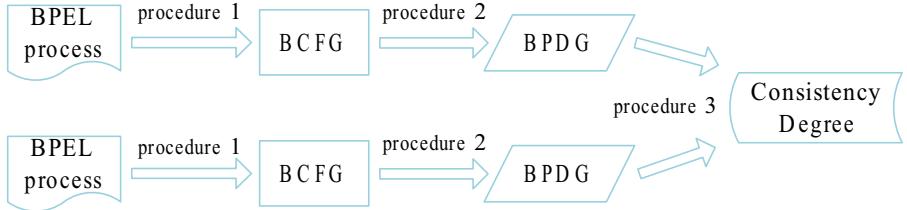


Fig. 3. The framework of our consistency analysis approach

We show the advantages of our approach through BPEL processes of the travel agency. We make some adjustments and obtain these processes, according to the travel agency process in [6], [10]. As the case has the characteristics of the general BPEL process, it is applied to many publications. We first introduce the background of the travel agency as followed: based on the booking information of airline and hotel, the travel agency process can invoke the airline process and hotel process and send the confirmed information to the user.

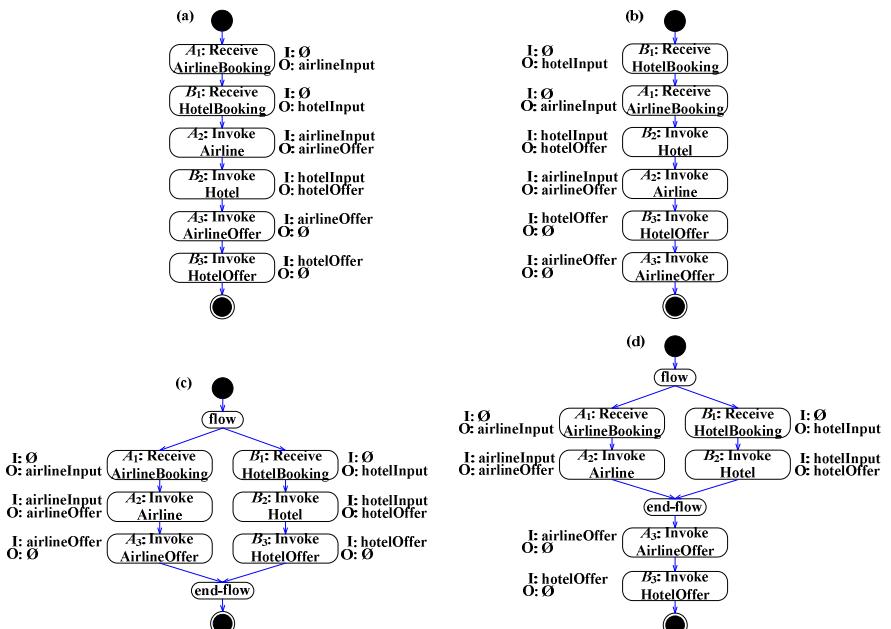


Fig. 4. The four variants of BCFG from BPEL process of the travel agency

Fig.4 is the four variants for the process model of the travel agency service on the same abstract level. Obviously, the differences between models (a) and (b) are that the execution order of the corresponding activities is changed. However, the order structures in the models (a) and (b) are transformed into the concurrent structure for the corresponding activities in the model (c). For the model (d), some corresponding activities are orderly executed, and the other are concurrently executed. We can select a pair from the four process models at random to calculate consistency degree based on trace equivalence and behavioral profiles respectively. For example, only one trace ($A_1 \rightarrow B_1 \rightarrow A_2 \rightarrow B_2 \rightarrow A_3 \rightarrow B_3$) in the model (a) can be mirrored in the model (c) that it contains twenty traces. According to [3], the consistency degree based on trace equivalence equals $P_T \sim = \frac{1+1}{1+20} \approx 0.0095$. Each model consists of six aligned activities, then, the numbers of the consistent activity pairs are $|T| \times |T'| = 36$ for both models. An analysis of models (a) and (c) reveals that each model contains eighteen corresponding activity pairs that are inconsistent based on behavioral profiles. That is, these corresponding activity pairs with the reversed pairs have different behavioral relations in both models. Therefore, the consistency degree based on behavioral profile is $P_P \sim = \frac{18+18}{36+36} = 0.5$.

Fig.5 shows four BPDGs that are respectively from the corresponding process models of Fig.4. As there is not selective structure in these BPEL processes, exclusion relation is not considered. We are able to realize that control and data dependences between all corresponding activity pairs are consistent for each pair of BPEL processes in Fig.5. Thus, the four variants of the travel agency process are completely consistent on the same abstract level.

Fig.6 is a histogram that we check consistency degree based three approaches respectively for six pairs of aligned processes. Generally speaking, if there are more or

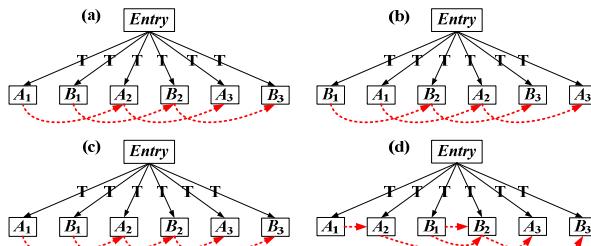


Fig. 5. Each pair of BPEL processes is completely consistent

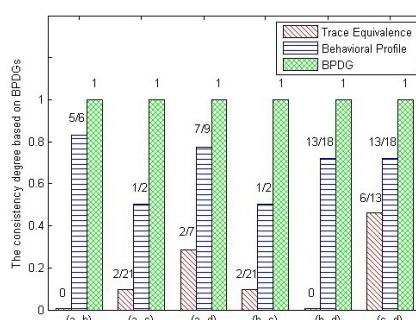


Fig. 6. The consistency degrees based on three approaches

less deviations between BPEL processes, these deviations can fall into two situations. In the first situation, the corresponding activity pairs from two BPEL processes have the same control flow, but the dependence relations are inconsistent. An example is that there is data dependence in an activity pair of a BPEL process, but the dependence relation does not exist in the corresponding activity pair of the other BPEL process. The approaches based on trace equivalence and behavioral profiles cannot identify the slight inconsistency. It only returns a false result to users. On the contrary, our approach can identify these inconsistent activity pairs and provides a more reasonable consistency degree. In the other situation, two BPEL processes have different control flows, but their dependence relations are normally consistent. For instance, an activity pair of a BPEL process is the strict order relation, and the corresponding activity pair of the other BPEL process is the concurrency relation. If there are no dependences in the two activity pairs, the consistency degrees based on trace equivalence and behavioral profiles respectively are lower than the consistency degree based on BPDGs. Our examples belong to the second situation. From Figure 6, we can conclude that it is appropriate to assess consistency degree of BPEL processes based on BPDGs.

4 Evaluation

In this part, we first introduce some implementation issues of our prototype tool, and then demonstrate the application of our approach in practice. Finally, we analyze the threats of our experiment.

4.1 Implementation

We have implemented a proof-of-concept tool that can automatically calculate the consistency degree between BPEL processes. The input of this tool is two BPEL processes and its output is the behavioral consistency degree of the two BPEL processes. In the implementation, the two BPEL processes are first transformed into two BCFGs, and each node in the BCFGs is associated with some relevant information, such as the name of activity, the variables of input and output, partnerLinks, etc. Then, BPDGs of the two BPEL processes are derived by control, data, and asyn-invocation dependence analysis based on the BCFGs. We use the matrixes to represent BPDGs, which facilitates the calculation of the consistency degree of the two BPEL processes. The matrixes and BPDGs are displayed in the picture box for diagnosis.

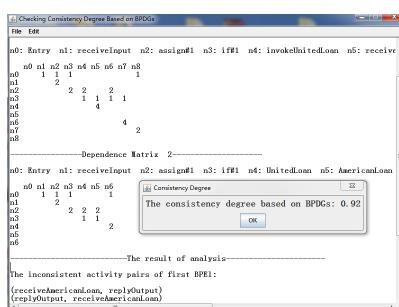


Fig. 7. The snapshot of our prototype tool

Fig.7 shows a snapshot of our tool. We can see that all dependence relations are demonstrated in the form of matrix for each BPEL process. For convenience, we assume that 1, 2 and 4 denotes the control, data and asyn-invocation dependences respectively.

4.2 Experiments

In order to evaluate the applicability of our approach in practice, we apply our prototype tool to analyzing consistency degrees of some BPEL processes borrowed and adapted from the Oracle BPEL Process Manager Samples. These BPEL processes are typical and cover a wide range of BPEL concepts and structures. We select some appropriate processes, and make some adjustments to obtain several variations.

In our experiment, we calculate the consistency degrees of these BPEL processes based on trace equivalence, behavioral profiles and BPDGs respectively. Fig.8 (a) describes the distribution of the consistency degrees based on BPDGs against based on trace equivalence. Apparently, our approach shows a high alignment, when the consistency degree based on trace equivalence equals 0, that is, no trace is mirrored. With a high corresponding trace, the consistency degree based on BPDGs might be higher. Once the consistency degree based on trace equivalence equals 1.0, the two BPEL processes are also completely consistent based on BPDGs. Fig.8 (b) shows the distribution of the consistency degrees based on BPDGs against based on behavioral profiles. Due to all alignments with at least two correspondences, the consistency degrees by using the two approaches cannot equal 0. We can find that the consistency degrees based on BPDGs are concentrated in higher values even equals 1.0. When the consistency degree based on behavioral profiles is more than 0.9, our approach can obtain a consistency degree approaching 1. Obviously, BPEL processes with a high alignment can get a high consistency degree by using our approach even if the consistency degree is somewhat low by using trace equivalence and behavioral profiles.

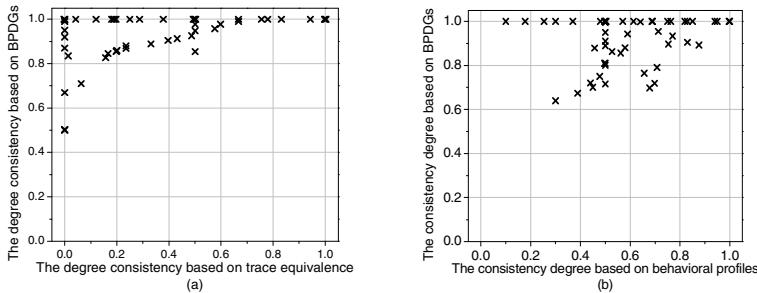


Fig. 8. The values analysis of consistency based on three approaches

4.3 Threats to Validity

The selection of process pairs could introduce a potential threat to validity. We already mentioned that we did not try to apply a random sampling, but rather relied on those variants of the same BPEL process. These BPEL processes are somewhat simple in the dimension and structure and they are propitious to our experiment. This implies a certain threat to representativeness. Additionally, despite the members that

construct these BPEL processes belong to the same research group with a similar professional background, everyone understands BPEL processes differently. This might make some deviation for deciding on the degree of consistency.

The paper chooses some appropriate BPEL processes to illustrate the feasibility of our approach and the results show that it is promising. However, one major concern is the external validity of our approach, namely, whether or not our approach is applicable to other BPEL processes. With this consideration in mind, we plan to apply our approach to more large-scale, evolving BPEL processes in the future.

5 Discussion

Our consistency measurement is defined on the activity relations (exclusiveness and different kinds of dependences) in BPEL processes. In practical uses, however, different kinds of activity constraints may lead to different effects for the consistency degree. For example, control dependence might be regarded as more important than data dependence in consistency measurement. In these situations, we should consider the weights of different kinds of activity constraints in the definition of consistency measurement. Nevertheless, it is not an easy job because the weights are somewhat subjective and some domain knowledge also needs to be considered. We plan to study this problem in our future work.

It is noted that the consistency degree is highly related to the process alignment. Our consistency analysis approach first automatically determines the aligned activity pairs in two BPEL processes, and then calculates the consistency degree of the two BPEL processes. Therefore, the consistency result will be affected if the aligned activity pairs are wrong. To address this limitation, when the consistency degree of BPEL processes is below one (e.g., 0.8), the aligned activity pairs and inconsistent activity pairs of the two BPEL processes can be provided to users. Users can leverage their domain knowledge to make a decision whether or not the alignment is reasonable and inconsistencies are tolerable. Another limitation of our approach is that it only covers some basic structures and elements of WS-BPEL 2.0. Some advanced structures are not fully supported yet, e.g., <for each>, <link>, etc. Thus, our approach is now only applicable to a subset of BPEL processes.

The consistency measurement is a first attempt to define the consistency degree of BPEL processes. However, it is unknown how the experts of business processes assess the consistency between BPEL processes, and which kind of measurement criterion they perceive to be appropriate. In the future, an empirical study based on statistical hypothesis testing is needed to show the appropriateness of our approach.

6 Related Work

In this section, we review two related areas of research: the consistency and similarity of workflow models, the substitutability and consistency of BPEL processes.

Weidlich et al. propose in [3] a consistency measurement of process models based on the notion of behavioral profiles which capture the essential behavioral constraints of process models. The approach is wonderful to assess the consistency between process models. Compared with the approach based on trace equivalence, our approach

can provide a consistency degree ranging from 0 to 1 rather than a “true/false” answer. However, both the approaches based on trace equivalence and behavioral profiles work at the level of abstract process models and are not appropriate to determine the consistency degrees of BPEL processes.

The concept of similarity is similar to the definition of consistency in the software engineering. Determining the similarity degree between process models is important for management, reuse, and analysis of workflows. However, how to retrieve the similar models efficiently from a large model repository is a challenge. Most of the existing works of business process similarity are based on the control flow graphs of the process. Various notions of edit distance are used for deciding on the differences of two processes [11], [12], [13]. Recently, different approaches that measure the similarity between models based on their behaviors have been proposed. One metric is built from five elementary similarity measures that are based on behavioral profiles and the Jaccard coefficient [14]. The proposed similarity measure is based on the notion of “principal transition sequences”, which aims to provide an approximation of the essence of a process model in [15]. van Dongen et al. use causal footprints as an abstract representation of the behavior captured by a process model. Based on the causal footprint derived from two models, their similarity based on the established vector space model from information retrieval can be calculated. [16].

As the first section discussed, there are various researches that are related to the substitutability of BPEL processes and process selection based on the behavior of BPEL processes. In terms of behavior-based process selection, it can take a directed graph as control flow model of BPEL process. Then, the problem of BPEL process selection converts to the problem of an alignment of the directed graphs [11]. When there are not consistent BPEL processes to meet the user’ requirements in BPEL process repository, it can select similar processes to the user [17]. Song et al. propose a BPDG-based approach for the substitutability analysis of WS-BPEL processes [6]. This approach considers both synchronous and asynchronous communication mechanisms into account, and thus is promising in practice.

7 Conclusion and Future Work

In this paper, we propose a novel behavioral consistency measurement and corresponding approach to quantitatively determining the behavioral consistency of BPEL processes based on their BPEL program dependence graphs. We implement our approach in a prototype tool. To show the applicability of our approach in practice, we apply our approach to analyzing consistency between the real-life BPEL processes.

Our current work still has some limitations which are discussed in Section 5. In the future, we will try to address these limitations to improve the applicability and effectiveness of our approach and also leverage activity constraints captured in BPDGs to quantify the similarity between BPEL processes.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (No. 61202003), the Specialized Research Fund for the Doctoral Program of Higher Education (No. 20113219120021), the Major Research Project of Jiangsu Province (No. BK2011022), and the State Key Laboratory of Software Engineering (No. SKLSE2012-09-05).

References

1. WS-BPEL 2.0 Specification (2007), <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>
2. van Glabbeek, R.J.: The linear time – branching time spectrum. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 278–297. Springer, Heidelberg (1990)
3. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. *IEEE Transactions on Software Engineering* 37(3), 410–429 (2011)
4. IEEE, Standard glossary of software engineering terminology (1990)
5. Song, W., Ma, X.X., Cheung, S.C., Hu, H., Yang, Q.L., Lü, J.: Refactoring and publishing WS-BPEL processes to obtain more partners. In: 2011 IEEE International Conference on Web Services, pp. 129–136 (2011)
6. Song, W., Tang, J.H., Zhang, G.X., Ma, X.X.: Substitutability analysis of WS-BPEL services. *China Science: Information Science* 42(3), 264–279 (2012) (in Chinese)
7. Nanda, M.G., Chandra, S., Sarkar, V.: Decentralizing execution of composite Web services. In: OOPSLA 2004, vol. 39(10), pp. 170–187 (2004)
8. Ferrante, J., Ottenstein, K.J., Warren, J.D.: The program dependence graph and its use in optimization. *TOPLAS* 9(3), 319–349 (1987)
9. Ramackers, G.J.: Integrated Object Modelling. PhD dissertation, Leiden University (1994)
10. König, D., Lohmsnn, N., Moser, S.: Extending the compatibility notion for abstract WS-BPEL processes. In: Proc. of WWW 2008, pp. 785–794 (2008)
11. Corrales, J.C., Grigori, D., Bouzeghoub, M.: BPEL processes matchmaking for service discovery. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 237–254. Springer, Heidelberg (2006)
12. Li, C., Reichert, M., Wombacher, A.: On measuring process model similarity based on high-level change operations. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 248–264. Springer, Heidelberg (2008)
13. Mahmud, N.M., Sadiq, S.W., Lu, R.: Similarity matching of business process variants. In: Proc. of the Tenth Int. Conf. on Enterprise Information Systems, ISAS-2, pp. 234–239 (2008)
14. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity – A proper metric. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 166–181. Springer, Heidelberg (2011)
15. Wang, J., He, T., Wen, L., Wu, N., ter Hofstede, A.H.M., Su, J.: A behavioral similarity measure between labeled Petri nets based on principal transition sequences. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010, Part I. LNCS, vol. 6426, pp. 394–401. Springer, Heidelberg (2010)
16. van Dongen, B.F., Dijkman, R., Mendling, J.: Measuring similarity between business process models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
17. Grigori, D., Corrales, J.C., Bouzeghoub, M., Gater, A.: Ranking BPEL processes for service discovery. *IEEE Transactions on Services Computing* 3(3), 178–192 (2010)

Efficient Complex Event Processing under Boolean Model

Shanglian Peng¹, Tianrui Li², Hongjun Wang²,
Jia He¹, Tiejun Wang¹, Fan Li¹, and An Liu¹

¹ School of Computer, Chengdu University of Information Technology,
Chengdu 610225, China

² School of Information Science and Technology, SouthWest Jiaotong University,
Chengdu 610031, China
`{psl,jejia,tjw,lifan,liuan}@cuit.edu.cn,`
`{trli,wanghongjun}@swjtu.edu.cn`

Abstract. Existing evaluation techniques for complex event processing mainly target sequence patterns. However, boolean expression represents a broader class of patterns and is widely used to model complex events in practice. In this paper, we study efficient evaluation techniques for complex events modeled as boolean expressions. We propose an evaluation approach based on pseudo-NFA. The efficiency of pseudo-NFA approach is investigated and its optimization techniques are presented. We show that the proposed techniques are theoretically sound and demonstrate by extensive experiments that they perform considerably better than existing alternatives.

Keywords: Complex Event Processing, Boolean Model, NFA, data stream, RFID.

1 Introduction

Sensing technologies such as Radio Frequency IDentification(RFID) have been widely used in modern monitoring applications, such as logistics and supply chain management, asset tracking and activity monitoring. Their successful deployments require primitive events to be correlated and transformed into meaningful complex events that reach a semantic level appropriate for end applications.

The technique of complex event processing(CEP) has been extensively studied in the context of active databases [1] and message-based systems [2] in the past. However, these methods are not suitable to process modern monitoring systems due to huge-volume/velocity and complicated event attribute constraints.

Existing CEP approaches [3][4][5] mainly focus on sequence patterns. They assume the events are ordered by their timestamps and exploit NFA to achieve high-performance processing. Even though sequence represents an important class of complex event pattern, boolean pattern is more general and has many applications:

Example 1. In an emergency alarming application, an event detection is usually based on a fusion of multiple sensed values of different sensors. For instance, the

occurrence of fire should satisfy the conditions such as $temperature > 100^{\circ}C$ and $smoke > 100mg/L$. Such composite events are represented by boolean expressions.

Example 2. In a RFID-monitored retail store, the manager is interested in the customers that visit three places(A, B and C) within one hour in whatever sequence. This scenario can be described by the following complex event: (a and b and c) satisfying ($a.location = A$) and ($b.location=B$) and ($c.location = C$) and ($a.customerID = b.customerID=c.customerID$) within one hour. The attribute *location* represents an event's occurrence place and *customerID* identifies an event's corresponding customer.

Example 3. In stock trading, an abnormal fluctuation on the prices of a stock can be detected by complex event patterns. For instance, consider the scenario that stock A's price rises by more than 10 percent within 10 minutes and falls more than 5 percent within the next 10 minutes. It can be described by the following complex event: (a_1 and a_2 and a_3) satisfying ($a_1.time < a_2.time \leq a_1.time + 10minutes$) and ($a_2.time < a_3.time \leq a_2.time + 20minutes$) and ($a_2.price \geq a_1.price \times 110\%$) and ($a_3.price \leq a_2.price \times 95\%$). In the specification, a_i represents a trading event on stock *A*, the attribute *time* is its trading time and *price* is its trading price.

In this paper, we study the issues of processing complex events modeled as boolean expressions. Our solution is based on pseudo-NFA. Compared with the traditional NFA which tracks the state transfer of complex events, pseudo-NFA tracks the state transfer of primitive events. As a pseudo-NFA may have multiple valid state transfer routes, we further study the optimization problem of how to identify the state transfer route which minimizes the overall evaluation cost.

The contributions of this paper are summarized as follows:

1. We propose an efficient evaluation approach based on pseudo-NFA for the general boolean model.
2. On the problem of pseudo-NFA optimization, we present a dynamic programming algorithm to compute the optimal evaluation roadmap for tree-structured pattern.
3. Our extensive experiments on synthetic and simulated sensing data demonstrate that the proposed techniques perform considerably better than existing alternatives.

The rest of this paper is organized as follows. Related work is discussed in Section 2. We introduce the boolean complex event model in Section 3. The general pseudo-NFA evaluation approach and its optimization is described in Section 4 and Section 5 respectively. The experimental study is presented in Section 6. Finally, we conclude this paper with Section 7.

2 Related Work

Complex event processing has been previously studied in active databases and message-based systems [1]. However, the presence of attribute value comparison constraints and sliding windows in the complex event specifications for modern monitoring applications makes these traditional processing frameworks no longer applicable. Data structures, such as tree[1], finite automata[6], cannot flexibly accommodate the necessary extensions to traditional complex event specifications.

Early work on complex event processing for modern monitoring applications include Berkeley's HiFi project [7] and Siemens RFID middleware [8]. In the HiFi system, complex event processing was used to support efficient aggregation of data generated by distributed interceptors. Complex event processing in Siemens middleware used the graph computation model to transform raw RFID readings into semantic data. Unfortunately, neither HiFi nor Siemens RFID middleware provides the necessary techniques to optimize huge-volume event processing.

In [3], Eugene et al. proposed a declarative specification language SASE for complex events and presented a corresponding evaluation framework based on NFA. Their work addressed the optimization issues of huge-volume event processing. The Cayuga project at Cornell [9] also defined its own complex event language and provided a variant of NFA for event evaluation. Note that the effectiveness of the SASE and Cayuga techniques depends on the sequential structure of event pattern. They do not support boolean event pattern. More recently, [10] proposed a CEP system called *ZStream* which is supposed to support efficient complex event processing. Its tree-based query plans mainly target sequence patterns and can not be directly applied on boolean model either.

3 Boolean Complex Event Model

A primitive event is supposed to have the schema of $(type, id, attr_1, \dots, attr_k)$, where $type$ denotes event type, id is event identifier, and $attr_i$ ($1 \leq i \leq k$) denotes the value of the i th attribute of an event. For instance, for an RFID event, event type is the reader location. Tag identifier and reading timestamp as well as other information are all its event attributes. Some of the denotations used by this paper are summarized in Table 1.

Definition 1. A boolean complex event pattern is defined in the following recursive way:

- (1) Any singular event type is a valid pattern.
- (2) $bop(E_1, E_2, \dots, E_m)$ is a valid pattern, where bop denotes one of three boolean operators (*and*, *or* and *not*) and each E_i represents a valid pattern.

Besides pattern, complex event specification may involve attribute value comparisons between events and have a time window within which all events are supposed to occur. Formally, a boolean complex event is specified by the following syntax:

Table 1. Symbol Denotations

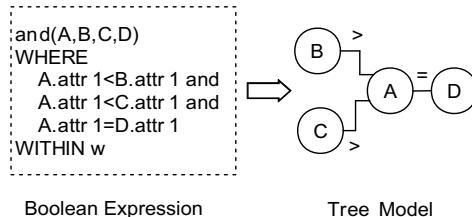
Denotation	Explanation
E_i, A, B, C, D, E	Event Types
P, P', P_i	Boolean Event Patterns
$e_i, e_{jk}, a_i, b_i, c_i$	Primitive Event Instances
$attr_i$	An Event Attribute
$type(e_i)$	The Event Type of e_i
$bop()$	A Boolean Operator, <i>and</i> , <i>or</i> , or <i>not</i>
$func_k(E_i.attr_j)$	A Function on E_i 's Attribute $attr_j$

PATTERN boolean-pattern
 WHERE attribute_value_comparison (optional)
 WITHIN time_window (optional)

In a boolean complex pattern, an attribute value comparison has the format of ($func_1(E_i.attr_k) op func_2(E_j.attr_k)$), in which op is one of the six comparison operators ($<$, $>$, $=$, \leq , \geq , and \neq). We suppose that the comparison conditions can only be specified on the same attribute, and multiple comparisons are connected by boolean operators. A time window can be specified on a pattern connecting the events by the *and* operator. Semantically, the specification ($and(E_1, E_2, \dots, E_k)$ **WITHIN** w) detects a complex event (e_1, e_2, \dots, e_k) , where $type(e_i) = E_i$ ($1 \leq i \leq k$) and $\max\{e_i.timestamp\} - \min\{e_i.timestamp\} \leq w$.

On complex event specification, boolean is a more powerful expression model than sequence. A sequence complex event specification $\{SEQ(E_1, E_2, \dots, E_k)\}$ **WITHIN** w can be transformed into an equivalent boolean one:

PATTERN *and*(E_1, E_2, \dots, E_k)
 WHERE $E_1.timestamp < E_2.timestamp < \dots < E_k.timestamp$
 WITHIN w

**Fig. 1.** An Example Boolean Complex Event Model

The basic boolean transformation rule:

$$and(E_1, or(E_2, E_3)) = or(and(E_1, E_2), and(E_1, E_3))$$

implies that any boolean expression can be transformed into the format of $or(E_1, E_2, \dots, E_k)$, where E_i ($1 \leq i \leq k$) contains only the *and* and *not* operators. It is observed that each E_i within the *or* boolean operator can be evaluated independently. We model an **and** boolean expression consisting of singular event types by a **graph**, in which each node represents an event type and an edge between two nodes corresponds to an attribute value comparison constraint. An example of boolean complex event specification and its corresponding tree model is shown in Figure 1.

4 Pseudo-NFA Evaluation Approach

In this section, we present the evaluation approach for the case where the attribute value comparisons between events of a boolean pattern involve only one attribute. Our solution is based on pseudo-NFA. We first define pseudo-NFA, and then present the evaluation approach for tree boolean pattern.

4.1 Pseudo-NFA

Pseudo-NFA is defined on tree boolean pattern. It characterizes a primitive event's state evolvement process from its initial state to its final state. A primitive event's state corresponds to its matching status on a complex event pattern. Given a tree pattern P , the possible states of an A event a_i include any connected sub-tree of P' rooted at the node A . An event a_i is in the state of P' if and only if there is at least one complex event instance which includes a_i and matches P' . For instance, on the tree pattern $P=(A-B-C)$, an A event can be in the state of $(A), (A,B)$ or (A,B,C) .

Figure 2(a) show the pseudo-NFA construction process of a specific event type. An example is also presented in Figure 2(b). Since a primitive event can be a component event in several complex event instances, it may be in different states simultaneously. For instance, in Figure 2(b), a B event b_i can be in both states of (BCD) and (BCE) . The event b_i may be correlated with two C events, c_j and c_k , whose states are (CD) and (CE) respectively.

4.2 Evaluation Approach for Tree Boolean Pattern

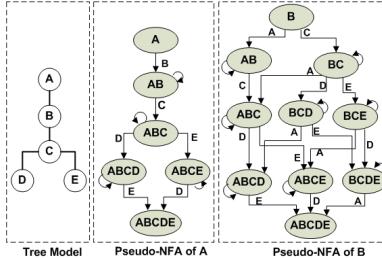
For tree boolean pattern evaluation, we suppose attribute value comparison is represented as $(E_i.attr_k op E_j.attr_k)$, where op is one of the five comparison operators ($<$, $>$, $=$, \leq and \geq), and only one comparison constraint is specified on each edge on tree pattern T . Additional symbol denotations used by this section are summarized in Table 2

Basic Pseudo-NFA Approach. We store the same type of events in a B+-tree ordered by their attribute values. Each entry in the B+-tree has a data structure recording its corresponding event's state information. Suppose a event node A is directly connected with n nodes, E_1, E_2, \dots, E_n , on a tree pattern P . A newly arriving event a_i is processed by the following steps:

Algorithm 1: Pseudo-NFA_Construction(EventType A)

- $S(A) = \{A\}$; (set initial state)
- While ($S(A) \neq \{P\}$)
 - For (each state $P' (\neq T)$ in $S(A)$)
 1. Delete P' from $S(A)$;
 2. For (each edge $E_i - P'$ on P)
 - (a) Record the state transfer ($P' \rightarrow (E_i, P')$);
 - (b) Insert (E_i, P') into $S(A)$;

(a) Pseudo-NFA Construction Algorithm



(b) Pseudo-NFA Examples

Fig. 2. Pseudo-NFA Construction Process and Illustration**Table 2.** Symbol Denotations used in Boolean Pattern Evaluation

Denotations	Explanation
P_i	Event patterns or event states
$\Pi_{P_j} P_i$	Project the pattern P_i on P_j
$ES, size(ES)$	Event stream and its Size
$S(a_i), S(e_{jk})$	Event states set of a_i or e_{jk}
$num(E_i), num(E_i-E_j), num(P)$	Event instance numbers of a pattern

Step1. Insert a_i into the B+-tree of A and determine a_i 's state.

Step2. Update the events states directly or indirectly correlated with a_i .

Step3. If a_i is in its final state, generate and output the corresponding complex event instances including a_i .

Step1 can be accomplished by performing search operations on the relevant B+-trees. We describe how to perform state transfer on the event a_i . We use P_j to denote the resulting subtree rooted at E_j (for $1 \leq j \leq n$) after cutting all the edges of A on the tree pattern P and use $S(j)$ to denote the set of distinct states of the E_j ($1 \leq j \leq k$) events correlated with a_i . If an E_j event e_{jk} correlated with a_i has a state of P_{jk} , a_i should be in the state of $((A) \cup (P_j \cap P_{jk}))$. Generally, the states of a_i are computed as:

$$(A) \times \Pi_{P_1} S(1) \times \Pi_{P_2} S(2) \cdots \times \Pi_{P_n} S(n)$$

where $\Pi_{P_j} S(j)$ projects each state in $S(j)$ on the subtree P_j .

Step2 propagates the event a_i 's new states to other events directly or indirectly correlated with a_i . It repeatedly performs state transfer on the events directly correlated with an event whose states have transferred. Consider an E_j event e_{jk} directly correlated with a_i .

In **Step3**, the result is output in a recursive way: (1)for each edge $(A-E_j)$ (for $1 \leq j \leq n$) on P , the algorithm identifies all the E_j events in the state of P_j ; (2)for each e_{jk} event, compute its complex event instances matching P_j ; (3)the event a_i is correlated with the complex event instances computed in the previous step to generate the final complex event instances matching P .

5 Pseudo-NFA Optimizations

A pseudo-NFA of an event type A on a tree pattern P specifies all possible state transfer routes taken by an A event to reach its final state (P) as shown in Figure2(b). In this section, we propose an approach to optimize the event state transfer roadmap of a pseudo-NFA. It partitions P into multiple (probably nested) subtrees, each of which corresponds to an evaluation block. For instance, in the first case of Figure 3, P_1 represents the block $(A-B)$ and P_2 represents the block (P_1-C) . In the second case, the node C is grouped with node D to constitute a new block P_3 . In both cases, there is only one state transfer route for the B events.

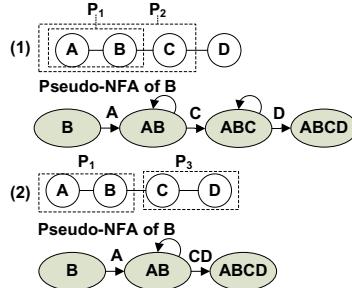


Fig. 3. Pseudo-NFA Optimization

For simplicity of presentation, we use $P=((A-B)-C)$, where $(A-B)$ is a block, as an example to illustrate the evaluation approach for tree pattern with defined blocks. Whenever a new C event c_i arrives, the algorithm first correlates c_i with the B events in the state of (A, B) . The algorithm then determines the states of c_i and output the results if necessary.

Whenever a B event b_i arrives, the algorithm first searches its corresponding A events. There are two possibilities:

- (1) No A event is correlated with b_i . In this case, the algorithm does not need to search C events according to pseudo-NFA.

(2) At least one A event is correlated with b_i . In this case, the algorithm should continue to search the C events correlated with b_i . If no C event is correlated with b_i , the algorithm sets b_i 's state to (A, B) . Otherwise, it sets b_i 's state to (A, B, C) and output the matching results.

Whenever an A event a_i arrives, the algorithm first searches its corresponding B events. If there is no B event correlated with a_i , the algorithm exits to process next event. Otherwise, it processes these B events sequentially and updates a_i 's states accordingly:

- (1) If the current B event b_j is in the state of (B) , the algorithm continues to search the C events correlated with b_j and updates b_j 's state accordingly.
- (2) If b_j is in the state of (A, B) , the algorithm stops the scanning. In this case, it does not need to search the C events correlated with b_j .

We now compare the evaluation efficiency of $P_1=(A-B-C)$ with that of $P_2=((A-B)-C)$. We use $\text{num}(P_i)$ to denote the total number of complex event instances matching P_i , by $|P_i|$ in this section. According to the pseudo-NFA evaluation approach described in Section 4.2, Step 1 of evaluating P_1 invokes totally $(|A| + 2|B| + |C|)$ searches while evaluating P_2 invokes totally $(|A| + |B| + |\Pi_B \text{Set}(AB)| + |C|)$ searches, where $\text{Set}(AB)$ denotes the set of complex event instances matching the pattern (AB) and Π is the *project* operator. Since $|\Pi_B \text{Set}(AB)| \leq |B|$, evaluating P_2 invokes no more searches than P_1 . Note that state transfer only needs to be performed at the B events because the event nodes A and C are leaves on P_1 and P_2 . On P_1 , the pseudo-NFA of B events has two state evolvement routes: $((B) \rightarrow (AB) \rightarrow (ABC))$ and $((B) \rightarrow (BC) \rightarrow (ABC))$. Its total number of executed state transfers is $(ns_b(AB) + ns_b(BC) + ns_b(ABC))$, in which $ns_b(P)$ represents the number of the B events that are in the state of P after the whole evaluation process. On P_2 , the pseudo-NFA for B events has only one state evolvement route: $((B) \rightarrow (AB) \rightarrow (ABC))$. Its total number of executed state transfers is $(ns_b(AB) + ns_b(ABC))$. The number of state transfers performed on P_2 is therefore less than or equal to that performed on P_1 .

5.1 A Dynamic Programming Evaluation Approach

Suppose a tree pattern P consists of two blocks (P_1-P_2), where node A and B are the boundary nodes in the blocks P_1 and P_2 respectively, we have the following lemma:

Lemma 1. *On a tree pattern P , if $P=(P_1-P_2)$, in which P_1 and P_2 are two separate blocks, is the optimal evaluation roadmap for P , the block structure within P_1 should also be the optimal evaluation map for P_1 . This is similarly true for P_2 .*

Lemma 1 assures the optimality of a sub-structure solution within a global optimal solution. The optimal evaluation roadmap for P therefore can be computed by a dynamic programming approach. The optimal evaluation roadmap for a subtree of a boolean pattern of size j is computed as follows:

P_j is partitioned into two subtrees whose sizes are less than j . For each such partitioning, compute the expected cost of its corresponding evaluation roadmap as described above. The one with the minimal cost is selected to be P_j 's optimal evaluation roadmap. Note that there are in total $(j - 1)$ possible partitions, each of which corresponds to deleting an edge on P_j .

6 Experimental Study

6.1 Experimental Setup

We have tested the performance of the proposed pseudo-NFA approaches on two datasets: synthetic sensing data and simulated RFID data.

Synthetic Sensing Data. Each synthetic primitive event has an attribute indicating its event type, an attribute recording its timestamp and two sensing attributes, $(attr_1, attr_2)$, which are used to specify the attribute value comparison constraints on tree patterns. Suppose there are four interesting event types (A, B, C, D) . The values of the attribute $attr_i$ (for $1 \leq i \leq 2$) are generated randomly within a preset interval.

Simulated RFID Data. We simulate customer's moving route in an RFID enabled retail store. Four locations are denoted by (A, B, C, D) . A customer's moving route is simulated by a walk on the complete graph consisting of four nodes. The elapsed time of moving between two locations are randomly set to be within an interval $[t_1, t_2]$. Supposing the four locations constitute a square in graph, we set the elapsed time intervals of moving between two locations as follows: $(AB) = (AC) = (BD) = (CD) = [9, 11]$, $(AD) = (BC) = [13, 15]$. The corresponding complex event pattern is specified as follows:

```
and(A, B, C, D)
WHERE A.cID = B.cID = C.cID = D.cID
WITHIN w
```

We measure evaluation efficiency by two metrics: CPU time, number of event searches which accurately quantify algorithm efficiency. We conduct the following comparative experiments to verify the effectiveness of the proposed pseudo-NFA approaches:

1. Pseudo-NFA vs Stream Join[11];
2. Pseudo-NFA Optimization, Basic Manner vs Block manner;

All the experiments are performed on a computer with Pentium(R) 4 CPU 2.8 GHz running on Windows XP and implemented in Visual C++.

6.2 Pseudo-NFA vs. Stream Join

We have implemented Stream Join algorithm in [11] and compared it with our method. On the synthetic event streams, we evaluate the chain tree pattern $P_1 = (A-B-C-D)$ in which the attribute value comparison constraints are specified

on the attribute $attr_1$ with $((A = B) \text{ and } (B < C) \text{ and } (C = D))$. Because of high volume and velocity of modern monitoring applications, we study these two evaluation approaches' scalability by measuring their performance variation with event stream size.

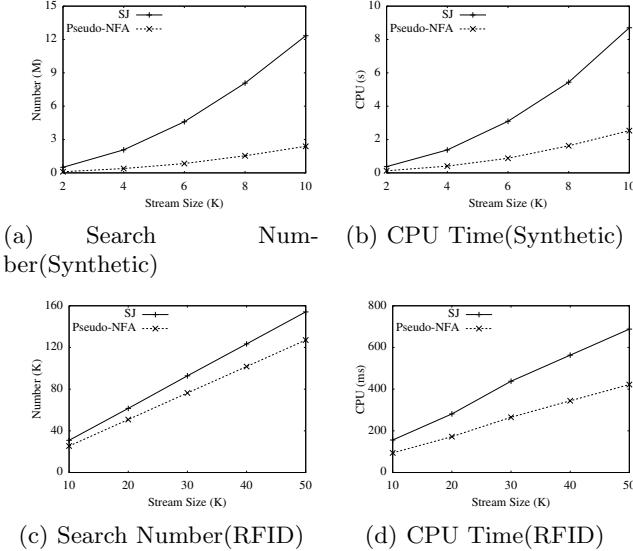


Fig. 4. Performance Comparison between Pseudo-NFA and Stream Join

The evaluation efficiency comparisons on the synthetic event streams and RFID event stream are presented in Figure 4 (a)-(d). In term of both search and event searches, the pseudo-NFA approach performs considerably better than stream join because search and traversing cost of stream join depend on the sizes of its intermediate evaluation results. The performance of pseudo-NFA is instead independent of such intermediate evaluation results.

6.3 Pseudo-NFA Optimization Techniques

In this subsection, we investigate the effectiveness of the roadmap optimization techniques in speeding up pseudo-NFA evaluation. The tested tree pattern on the synthetic event streams is $P_1=(A-B-C-D)$ with constraints $((A = B) \text{ and } (B < C) \text{ and } (C = D))$ specified on the attribute $attr_1$. On P_1 , we test four evaluation roadmaps : $P_{11}=(ABCD)$, $P_{12}=((AB)CD)$, $P_{13}=(((AB)C)D)$ and $P_{14}=((AB)(CD))$ respectively. To test pseudo-NFA optimization's sensitivity to edge selectivity on tree patterns, we use two types of synthetic event streams with $sel(A - B) = 0.05$ and $sel(A - B) = 0.2$, in which $sel(A - B) = \alpha$ means that only $(\alpha \times 100)$ percentage of the B events have their corresponding

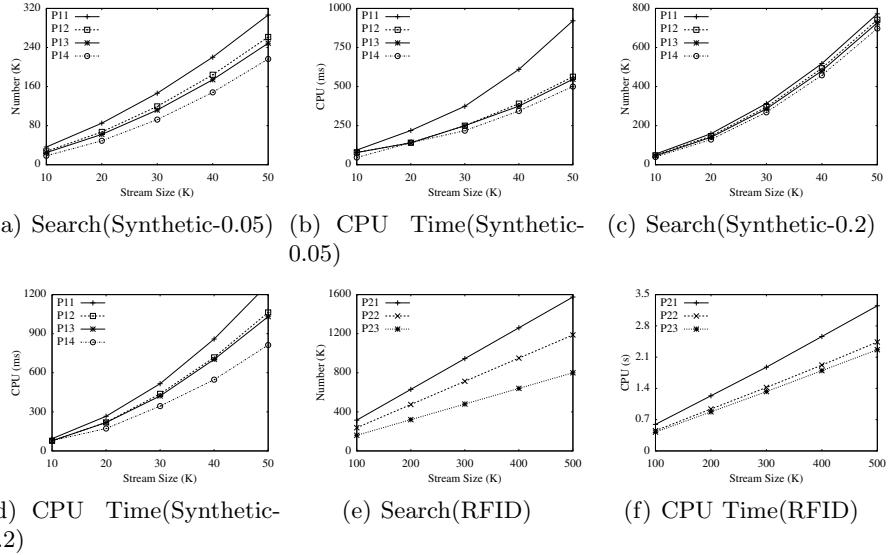


Fig. 5. Pseudo-NFA Roadmap Optimization

A events. The tested tree pattern on the RFID event stream is $P_2 = (A \cdot B \cdot C \cdot D)$. We tested three evaluation roadmaps : $P_{21} = (ABCD)$, $P_{22} = ((AB)CD)$ and $P_{23} = ((AB)(CD))$ respectively.

The evaluation efficiency comparisons on the synthetic event streams are presented in Figure 5 (a)-(d). It is observed that on all three cost metrics, the evaluation roadmaps P_{13} and P_{14} perform better than P_{12} , which in turn performs better than P_{11} .

The block evaluation manner of P_{14} therefore saves more search and traversing cost than P_{13} . It is also observed that the performance discrepancies in the case of $\text{sel}(A - B) = 0.05$ are more striking than in the case of $\text{sel}(A - B) = 0.2$. In the case of $\text{sel}(A - B) = 0.05$, the percentage of the B events satisfying $(A = B)$ is lower.

The evaluation results of P_2 on the simulated RFID event stream are presented in Figure 5 (e)-(f). Similarly, both P_{22} and P_{23} perform better than P_{21} on the three cost metrics. The attribute comparison constraints of P_2 are more rigorous than those of P_1 . The number of complex event instances matching P_2 is therefore less than that of complex event instances matching P_1 . Measured by percentage, the saved cost of search and event access on P_2 as a result of the block evaluation appears to be larger.

7 Conclusion

In this paper, we have proposed a cost model and the pseudo-NFA evaluation approaches for boolean complex events. We also present effective algorithms

to optimize the pseudo-NFA evaluation roadmap. Our extensive experiments have demonstrated the efficiency and effectiveness of the proposed pseudo-NFA approaches.

Several interesting problems remain to be investigated in future work. Firstly, the presented solutions to graph model and multi-dimensional attribute comparison evaluation are only preliminary. The CEP evaluation in these cases has potential to be further optimized. Secondly, aggregation operators have been shown to be important to expressivity of sequential complex event specification. How to incorporate them into boolean model and process them efficiently is also a challenging problem.

Acknowledgments. This work was mainly completed when the first author was studying in northwestern polytechnical university, thanks professor Chen Qun and Li Zhanhuai's supervision. This work is sponsored partially by National Natural Science Foundation of China (No. 61033007, No. 61003142 and No. 61262058) and Project(KYTZ201313) supported by the Scientific Research Foundation of CUIT.

References

- Chakravarthy, S., Krishnaprasad, V., Anwar, E., Kim, S.: Composite events for active databases: Semantics, contexts and detection. In: VLDB, pp. 606–617 (1994)
- Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2002)
- Wu, E., Diao, Y.L., Rizvi, S.: High-performance complex event processing over streams. In: SIGMOD, pp. 407–418 (2006)
- Wang, F., Liu, S., Liu, P., Bai, Y.: Bridging physical and virtual worlds: Complex event processing for RFID data streams. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 588–607. Springer, Heidelberg (2006)
- Agrawal, J., Diao, Y.L., Gyllstrom, D., Immerman, N.: Efficient pattern matching over event streams. In: SIGMOD (2008)
- Gehani, N.H., Jagadish, H.V., Shemueli, O.: Composite event specification in active databases: Model and implementation. In: VLDB, pp. 327–338 (1992)
- Franklin, M.J., Jeffery, S.R., Krishnamurthy, S., et al.: Design considerations for high fan-in systems: The hifi approach. In: Proceedings of the Conference on Innovative Data Systems Research (CIDR), pp. 290–304 (2005)
- Fusheng, W., Peiya, L.: Temporal management of RFID data. In: VLDB, pp. 1128–1139 (2005)
- Brenna, L., Demers, A., Gehrke, J., et al.: Cayuga: A high-performance event processing engine (demo). In: SIGMOD (2007)
- Mei, Y., Madden, S.: Zstream: A cost-based query processor for adaptively detecting composite events. In: SIGMOD (2009)
- Golab, L., Ozsu, M.T.: Processing sliding window multi-joins in continuous queries over data streams. In: VLDB (2003)

Bounding Trust under Uncertain Topology Information in Reputation-Based Trust Systems

Xi Gong¹, Ting Yu¹, and Adam J. Lee²

¹ Department of Computer Science, North Carolina State University
`{xgong2,tyu}@ncsu.edu`

² Department of Computer Science, University of Pittsburgh
`adamlee@cs.pitt.edu`

Abstract. Reputation-based trust evaluation relies on the feedback of an entity’s past interactions to estimate its trustworthiness for future behavior. Past work commonly assumes that all the information needed in trust evaluation is always available, which, unfortunately, often does not hold in practice, due to a variety of reasons. A key problem is thus to study the quality of trust evaluation in the presence of incomplete information. In this paper, we investigate techniques to bound the output of a trust evaluation under uncertain graph topologies of a reputation system. We present a rigorous formalism of the problem, and establish connections between uncertain topology information and missing feedback information through a property called *edge reducibility*. We show that the trust bounding problem can be efficiently solved if a trust function is edge reducible.

Keywords: Reputation, Trust management, Trust Bounding.

1 Introduction

Reputation is one of the major techniques to manage trust in decentralized systems. In a reputation system, after an interaction is completed, the participants may issue feedback that evaluates the services or behavior of others during the interaction. Before a new interaction is engaged, an entity may first assess the trustworthiness of potential participants based on the feedback issued by other entities. This trust evaluation process can be modeled as the application of a *trust function*, which takes as input the feedback information in a system and outputs a trust score for a target entity. Much work has been done on the design and efficient implementation of trust functions (see for example [2–7] for representative trust functions). However, most existing work assumes total access of information, i.e., an entity can always collect any feedback information when needed. Unfortunately, such an assumption often does not hold in practice, for example, due to access control restriction, unavailability of information holder or high cost of information collection.

A key question is then to assess the quality of trust evaluation given the uncertainty caused by missing information. Could the missing information cause

a big change to the trust computed only based on current available information? If so, the trust score is not reliable. We need consult other means to collecting more information and improve the quality of trust before we can make a sound security decision. This problem can be formally described as a trust bounding problem: given the range of possible values taken by missing information, what are the possible trust scores for an entity? In particular, we are interested in establishing the worst- and best-possible trust scores for a target entity. Such a bound can help us make better trust-related decisions. For example, suppose our policy is to only interact with entities whose trust scores are over 0.8. If the bounds of its trust score is [0.85, 0.95], then it is safe to interact with that entity, even if some information cannot be collected.

In general feedback information in a reputation system can be modeled as a weighted directed graph, whose nodes correspond to entities and whose edge weights indicate one entity's feedback rating to another. Given this graph representation, two types of missing information may exist. The first type is the missing weight information: the graph topology is totally known, but the weights associated with some edges are not certain. The second type is the missing topology information: it is unknown whether two entities had interactions before (not to say the feedback for such interactions). Mapping back to the graph representation, it means the existence of edges between them is unknown. In [1], we initiate the first effort to study the trust bounding problem under the first type of missing information. In this paper, we investigate techniques to bound trust when some topology information is unknown. Specifically, we make the following contributions. First, we give an exact formalism of the trust bounding problem under missing topology information through a general model of reputation-based trust assessment. Second, we identify an important property of trust functions called *edge reducibility* where the case of missing topology information can be reduced to the case of missing weight information. We show that if a trust function is edge reducible, then the trust bounding problem can be efficiently solved.

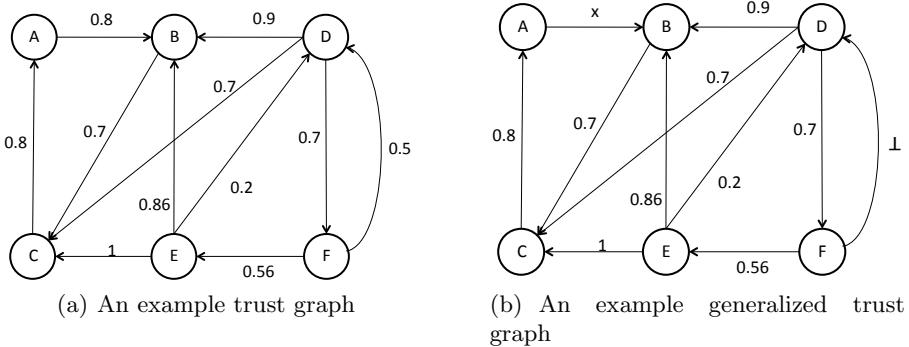
2 Reputation Systems and the Trust Bounding Problem

In general the interactions between entities in a decentralized system can be modeled through uni-directional transactions with a clear distinction between a service provider and a service consumer. A feedback is a statement issued by a consumer about a provider on the quality of service in a transaction. We denote a transaction as a four-tuple (c, s, r, t) , meaning that consumer c issues a feedback about provider s with a rating r at time t . For simplicity, we focus on ratings from binary positive/negative votes, which is consistent with the design of most existing trust functions.

Though multiple transactions may happen between a consumer c and a provider s , most trust functions in the literature are defined over an aggregation of the ratings of those transactions, e.g., the ratio of positive or negative transactions among all transactions between them. We call such an aggregation c 's *local trust* for s . The total information that are relevant to trust evaluation in a reputation system then can be modeled as a *trust graph* defined below.

Definition 1 (Trust Graph). Given a set of entities P and a totally ordered domain of weights W , a trust graph is a weighted directed graph $G = (P, E)$. The edge (c, s, w) in the set $E \in P \times P \times W$ encodes the local trust w that the consumer c has for the service provider s .

Figure 2 shows an example trust graph with six entities.



A *trust function* computes a trust score for an entity based on the feedback issued in a reputation system.

Definition 2 (Trust Function). Let \mathcal{G} be the set of all possible trust graphs, P be the set of all entities, and T be a totally ordered domain. A trust function $F : P \times P \times \mathcal{G} \rightarrow T$ is a function that takes as input a pair of entities u and v and a trust graph, and computes the extent of u 's trust in v .

Intuitively, given the feedback information stored in a trust graph, F computes the trustworthiness of v from u 's point of view. We call u and v the source and target respectively of a trust evaluation.

As mentioned in section 1, several types of missing information may exist in practice, which we categorize next. Let u and v be two entities in a trust graph $G(P, E)$. Then the relationship between u and v can be one of the following: (1) **Known weight:** it is known that v has provided services to u before, and u 's local trust in v is also known for certain. Reflecting in the trust graph, it means $(u, v) \in E$ with a known weight w ; (2) **Unknown weight:** we know that u used v 's services before and issued feedback, but cannot collect all the feedback. Thus we can only derive a range for the possible local trust of u to v ¹; (3) **No edge:** We know for sure that v did not provide services at all to u before, which means $(u, v) \notin E$; (4) **Unknown edge:** It is uncertain whether u has received v 's services before. Therefore, the edge (u, v) may or may not exist in

¹ For example, we know u has issued feedback for 10 transactions with v . However, we can only collect 6 of them with 5 positive ratings and 1 negative rating. Then the range of u 's local trust in v would be $[0.5, 0.9]$. In the extreme case, if we cannot collect any feedback from u to v , then the range would be $[0, 1]$.

G , and if it exists, we do not know its weight at all. Unknown edges also mean the topology of G is uncertain.

The definition of trust graphs can be generalized to incorporate the above types of incomplete information.

Definition 3 (Generalized Trust Graph). *Given a set of entities P and a total ordered domain of weights W , a trust graph is a weighted directed graph $G = (P, E)$, such that the weight associated with an edge (u, v) can be one of the following: (1) a weight $w \in W$, indicating u 's local trust to v is w ; (2) a weight variable with range $[l, u]$, indicating u 's local trust in v falls into $[l, u]$; and (3) a special variable \perp , indicating the edge (u, v) may or may not exist in G .*

Note that if (u, v) is not in a generalized trust graph, then it is certain that v did not provide services to u before. Figure 2 shows an example generalized trust graph. The edge from A to B is an unknown weight edge with weight variable x , and the edge from F to D is an unknown edge. If a trust graph does not contain any uncertain information, then we also say it is a *concrete trust graph*.

A generalized trust graph may correspond to a set of possible underlying trust graphs. For example, if an edge (u, v) is with a weight variable in the range $[0.5, 0.9]$, then assigning any value in the range as the weight will give us a different trust graph, which will result in different trust scores of a target entity when applying a trust function. Formally, we have:

Definition 4 (Graph Extension, \sqsupseteq). *Let $G(P, E)$ be a concrete trust graph and $G'(P', E')$ be a generalized trust graph. We say G extends G' , denoted $G \sqsupseteq G'$, if and only if (i) $P = P'$; (ii) $(u, v, w) \in E \rightarrow [(u, v, w) \in E' \vee (u, v, x_w) \in E' \vee (u, v, \perp) \in E']$, where x_w is a weight variable and w is in the range of x_w ; and (iii) if there is no edge from u to v in G' , then $(u, v, w) \notin G$ either.*

Intuitively, for an unknown weight edge in a generalized graph, we can assign a weight value in the range to make it concrete. Similarly, for the edge (u, v, \perp) , we can extend it by either assuming (u, v) does not exist or (u, v, w) exists with an arbitrary weight. It is easy to see that the concrete trust graph in figure 2 extends the generalized graph in figure 2.

Definition 5 (Trust Bounding Problem). *Given a generalized trust graph G' , a trust function F , a source entity s , and a target entity t , the trust bounding problem is to compute the pair $(\min_{G \sqsupseteq G'} F(G, s, t), \max_{G \sqsupseteq G'} F(G, s, t))$, denoted as $\text{Bound}(F, s, t, G')$.*

Clearly, if we treat a trust function as a black box (i.e., without leveraging any properties of a trust function), we can only enumerate all the possible extensions of a generalized graph to establish bounds of trust. This process is clearly computationally expensive even with modest amount of missing information. In our past work [1], we investigated the trust bounding problem with only unknown weight edges. The main technique is to study the monotonicity property of trust functions. Intuitively, if a trust function is monotonic to the weight variable x of an edge, then we only need to extend that edge with the boundary values of

x given its range, and obtain the trust bound. A nice property of monotonicity is that even for the case of multiple weight variables, as long as a trust function is monotonic to each of them, then the trust bound can be computed easily by examining the boundary combinations of these variables instead of exploring the whole possible space of extended graphs.

In this paper, we study how to bound trust under unknown edges. Our main technique is to establish the connection between unknown edges and unknown weight edges, and reduce the former to the latter so that techniques developed in [1] can be applied.

3 Trust Bounding with Unknown Edges

One may already observe that an unknown edge (u, v, \perp) can actually be divided into two cases. The first is a graph without the edge from u to v at all. And the other is to treat the edge as an unknown weight edge (u, v, x) , where x is in the range $[0, 1]$. So compared with the case with only unknown weight edges, it seems that we only need to compute trust for one more graph. This is true indeed if we only consider a single unknown edge. However, when we have multiple unknown edges, we have to consider all the possible combinations (with or without an edge) when extending a general trust graph, which would quickly yield a total space that is hard to enumerate. For example, if there are n unknown edges, then extending them will give us 2^n possible topologies for extended trust graphs.

The question is then whether we can avoid enumerating all the possible topologies. Our main technique is to study the relationship between the trust over a graph without the edge (u, v) and the trust bound over a graph with an edge (u, v, x) . If we can show that the trust in the former case always fall into the trust bound of the latter, then we can ignore the former case when bounding trust for a generalized graph. We call this property *edge reducibility*, which is formally defined as follows.

Definition 6 (Edge Reducibility). Let $G(P, E)$ be a generalized trust graph whose only missing information is an unknown edge $e = (u, v, \perp)$. Consider $G_1 = (P, E - \{e\})$ and $G_2 = (P, (E - \{e\}) \cup \{(u, v, x)\})$, where the range associated with x is $[0, 1]$. Given a trust function F and any pair of entities s and t , we say F is edge reducible if for all such G_1 and G_2 , $\text{Bound}(F, s, t, G_2)$ always covers $F(s, t, G_1)$.

Essentially G_1 corresponds to the case that we assume the unknown edge does not exist, and G_2 corresponds to the case that we assume the unknown edge exists but we do not know its weight. If a trust function is edge reducible, then the trust score over G is always within the trust bound over G_2 . In other words, the trust bound of G is the same as that of G_2 . In the above definition, we assume the only missing information in G is an unknown edge. What if G has multiple unknown edges or unknown weight edges? A nice property of edge reducibility is that it can be extended to cover such cases. We have the following theorems.

Theorem 1. Let $G(P, E)$ be a generalized trust graph with a single unknown edge $e = (u, v, \perp)$ and multiple unknown weight edges. Let $G_1 = (P, E - \{e\})$ and $G_2 = (P, (E - \{e\}) \cup \{(u, v, x)\})$, where the range associated with x is $[0, 1]$. If a trust function F is edge reducible, then $\text{Bound}(F, s, t, G_2)$ always covers $\text{Bound}(F, s, t, G_1)$ for all source s and target t .

Given the above theorem, we can further generalize it to the case with multiple unknown edges and unknown weight edges.

Theorem 2. Let $G(P, E)$ be a generalized trust graph with unknown edges d_1, \dots, d_n and unknown weighted edges e_1, \dots, e_k . Let $G_1 = (P, E - \{d_1, \dots, d_n\})$ and G_2 be the graph from G where all unknown edges are converted to unknown weight edges with range $[0, 1]$. If a trust function F is edge reducible, then $\text{Bound}(F, s, t, G_2)$ always covers $\text{Bound}(F, s, t, G_1)$ for all source s and target t .

4 Conclusion

In this paper we study the problem of trust bounding under incomplete topology information in reputation systems. After presenting a formal definition of the problem, we identify an important property of trust functions, namely, edge reducibility. This property enables us to reduce the trust bounding problem with unknown edges to the case of unknown weight edges, which has been studies in our previous work. In our future work, we would like to investigate mechanisms to integrate trust bounding techniques with the design of reputation systems so that it can be used to guide efficient information collection.

Acknowledgment. This research is partially supported by the NSF grant CCF-0914946.

References

1. Gong, X., Yu, T., Lee, A.J.: Bounding trust in reputation systems with incomplete information. In: CODASPY, pp. 125–132 (2012)
2. Jøsang, A., Ismail, R.: The Beta Reputation System. In: Proceedings of the 15th Bled eCommerce Conference (2002)
3. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The Eigentrust algorithm for reputation management in P2P networks. In: WWW Conference Series, pp. 640–651 (2003)
4. Lee, S., Sherwood, R., Bhattacharjee, B.: Cooperative peer groups in NICE. In: Proceeding of the INFOCOM 2003 (2003)
5. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 351–368. Springer, Heidelberg (2003)
6. Staab, E.S.: The Pudding of Trust Editor’s Perspective. IEEE Intelligent Systems 3(4), 19(5), 74–88 (2004)
7. Zhao, H., Li, X.: VectorTrust: trust vector aggregation scheme for trust management in peer-to-peer networks. The Journal of Supercomputing, 1–25 (2011)

A Novel Service Selection Based on Resource-Directive Decomposition

Dongming Jiang^{1,2}, Yuan Jiang³, Wuping Xie^{1,2}, and Zhen You^{1,2}

¹ State Key Laboratory of Software Engineering, Wuhan University, Wuhan

² Computer School of Wuhan University

³ School of Information Engineering, Nanchang University, Nanchang

jdm_nj_cn@yahoo.com.cn

Abstract. Since increasing services are deployed and published, service composition selection has became a crucial issue of Service-Oriented Computing (SOC). To address the issue, the paper proposed a novel selection algorithm based on the principle of divide and rule. We first construct a linear programming model to describe service selection problem; we then decompose the model into small subproblems by means of resource-directive decomposition technique. To obtain near-to-optimal resource allocation, we employ subgradient algorithm to search appropriate solution. To demonstrate our approach, we conducted a series of experiments to compare the performance of our approach with others.

Keywords: Service selection, resource-directive decomposition, QoS, Subgradient Method.

1 Introduction

The vision of SOC is to build loose-coupled, platform-independent business processes which fulfill users demands agilely. As the major implementation of SOC, Web services has gain increasing attention from both industry and academics, for it facilitate developers and users to construct workflows which tackle their needs conveniently. Meanwhile, the growing number of deployed services, which have similar function but different quality, poses a new challenge for users how to distinguish and select appropriate service. Hence, many approaches of Web service Composition Selection (WCS) [14][3] are proposed in recent years.

In general, the approaches of WCS are categorized into two groups: local selection strategy and global one. Using bottom-up strategy, the local approaches, which choose the combination of a optimal single service, fail to capture users end-to-end quality requirement. On the contrary, the global methods confront another obstacle resulting from the exponential searching complexity of candidate services. So, to address the above problem, we present a novel selection approach based on resource-directive decomposition. The main contributions of our work are threefold. Firstly, we build the mathematical model of service

selection from QoS decomposition-based perspective and convert it into Lagrangian dual problem. Secondly, we use subgradient method to find optimal strategy for Lagrangian dual problem, i.e., QoS allocation in the entire workflow. Finally, we conduct a series of experiments to illustrate the efficiency of our approach and compare it with other approaches.

2 Related Work

Since WCS is an important issue in SOA, extensive works have been concentrated on this area. Zeng et al. [13][14] developed an approach using integrate linear programming and critical path. The approach is effective at most medium-size problem, but fail to cope with large scale system due to its poor scalability. Ardagna et al. [4] also proposed a selection approach based on mixed integer linear programming (MILP) and present a negotiation mechanism when linear programming method cannot find the appropriate service. Similar to Zeng's work, the work of [4] also suffer from poor scalability due to its exponential time complexity.

Yu [12] presented two approaches for QoS-based service selection problem. The first methods modeled it as a multi-dimension multi-choice 0-1 knapsack problem, and then proposed WS-HEU algorithm to resolve it; the second method, based on graph approach, models the service selection problem as the constrained shortest path problem and applies heuristic algorithm to solve it. But the exponential size of the searching space is unavoidable obstacle for the graph-base algorithm.

Alrifai [2] proposed a scalable selection approach, which decomposes the original problem into small sub-problems. However, this approach is based on the directional derivative, which do not guarantee the convergence of the result strictly. Similar to Alrifai's work, our approach employs divide and conquer principle to deal with service selection problem; but we use the subgradient method to coordinate the interaction between the original problem and subproblems. Moreover, our approach converts WCS problem into Lagrangian dual problem, which is the major difference between our work and [2].

3 Our Works

Briefly, WCS problem can be stated as follows: Given a_i represents a activity (abstract service) and workflow $w = \langle a_1, \dots, a_n \rangle$ under end-to-end QoS constraints, the purpose of selection approach is to maximize users utility function $U(w)$. By using SAW technique [11] and assuming constraints equation is a linear equation, we construct a mathematical model of service selection by means of linear programming (LP) from the quality and activities perspective, respectively:

$$\begin{aligned}
\max U(w) &= c_1 \underbrace{\sum_{j=1}^n q_{j1}}_{q_1} + c_2 \underbrace{\sum_{j=1}^n q_{j2}}_{q_2} + \cdots + c_m \underbrace{\sum_{j=1}^n q_{jm}}_{q_m} \\
&= \underbrace{\sum_{i=1}^m c_i q_{1i}}_{a_1} + \underbrace{\sum_{i=1}^m c_i q_{2i}}_{a_2} + \cdots + \underbrace{\sum_{i=1}^m c_i q_{ni}}_{a_n}
\end{aligned} \tag{1}$$

subject to the constraints: $q_{1i} + q_{2i} + \cdots + q_{ni} \leq b_i, \forall i \in N$.

3.1 Resource-Directive Decomposition Strategy

Inspired by [2][9], we employ divide and conquer principle to tackle service selection. In other words, our approach transforms LP into n subproblem and obtains the solution of whole system from the sum of subsystems' solutions under the optimal resource allocation. Given a QoS allocation vector $q^k = \langle q_1^k, q_2^k, \dots, q_n^k \rangle$, the original problem 1 is converted into a new form P' as follows:

$$P' : \max_{q^k} U(w) = \begin{cases} \max_{q^k} \sum_{i=1}^n U(q_i^k) & s.t. \\ & \sum_{i=1}^n q_{ij}^k \leq b_j \end{cases} \tag{2}$$

Nevertheless, how to allocate the end-to-end QoS requirement to each activity optimally as so to maximize the overall utility is a NP-hard problem. Differ from other works [2][9], we utilize the resource-directive decomposition method to partition the overall QoS constraints. However, it exists two crucial issues about resource-directive decomposition. First, how to determine whether or not certain allocation plan q^k is optimal plan? Second, if q^k is not optimal, how to find more optimal allocation plan iteratively? Hence, we need a rigorous and iterative mechanism to find a near-to-optimal plan. For it is easy to implement and hold the rapid convergence feature, we employ the subgradient method [5][7], an iterative method for solving convex function optimization problems. In the first step, we convert equation 1 into its Lagrange-multiplier problem[6]:

$$\max U(w)_D = \sum_{i=1}^n \sum_{j=1}^m c_j q_{ij} + \sum_{j=1}^m \lambda_j (\sum_{i=1}^n q_{ij} - b_j) \tag{3}$$

Where $\lambda_j (\in R^n, \forall j)$ is the Lagrangian factor adjusted by the subgradient methods [10]. We then solve the dual problem using subgradient method iteratively:

$$\lambda_j^{k+1} = \lambda_j^k + s_j^k d_j^k \tag{4}$$

Where s_j is a positive step size and k is the iterative number. The subgradient d_j is the searching direction of algorithm, our approach updates subgradient iteratively according to:

$$d_j^k = \sum_{i=1}^n q_{ij}^k - b_j, \quad \forall j \in [1, m] \quad (5)$$

When $k > K$, where K is a constant given by users in advance, or $\|d_j\| = 0$, the subgradient algorithm terminates and our method calculates final solution.

4 Experiments and Discussion

In order to evaluate the performance of our method, we carried out a series of experiments and compare it with Zeng's work [14], for short LP, and Alrifai's work [2], for short DIST-HEU. We conducted two groups of simulations by varying the controlled parameters, including the number of activities and the number candidate service per activity. Moreover, we analyze our approach and others in term of the computational complexity and the accurate degree of those selection algorithms [9]. One metric is running time to measure how long the CPU time an algorithm need to achieve a solution, while approximate ratio measures the ratio of the utility value between different algorithms. In the article, we set the utility value obtained by LP method as the baseline of approximate ratio in order to evaluate other approaches.

The purpose of the first group of simulation is to investigate the time complexity when different approach get solution. As shown in Figure 1 and 2, Our approach and DIST-HEU can obtain a feasible solution faster than LP methods due to the former both employ QoS decomposition technique. The result in Figure 1 and 2 demonstrates our approach have better scalability for large scale LP problem. Compared to DIST-HEU, our approach consumes more at most 5 percent CPU time than DIST-HEU. We speculate the phenomenon is result from the computational complexity of subgradient method, because it needs to calculate Lagrangian dual problem $U(w)_D$ at each step.

In the second group of simulations, we measure approximate ratio of different approaches to evaluate the quality of returning solution. From Figure 3 and 4, we observe that our approach outperform than DIST-HEU in term of the quality of the result. Our approach can obtain average 92.5 per cent of the solution of LP, while DIST-HEU only achieve 86 per cent. Moreover, Figure 3 and 4 reflect the error of DIST-HEU increase with the growing of the dimension of problem, because DIST-HEU cannot guarantee the rapid convergence of the feasible solution.

As the discussion above, the experimental results demonstrate our solution outperforms LP in term of time and space complexity, and obtain better approximate solution than [2] under the same iterative number.

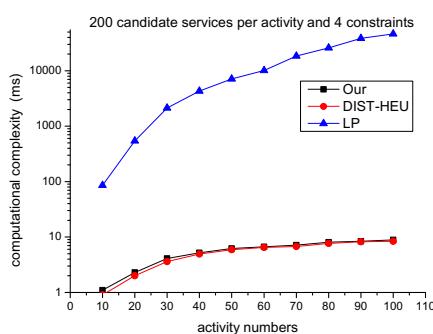


Fig. 1. Running time varying the number of activities

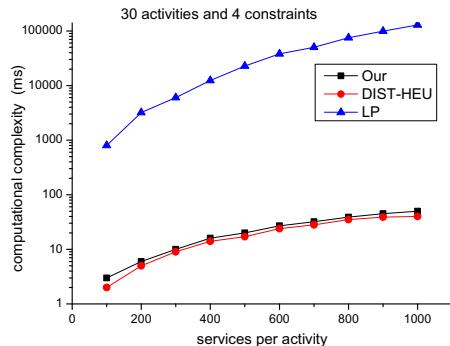


Fig. 2. Running time varying the number of candidate services per activity

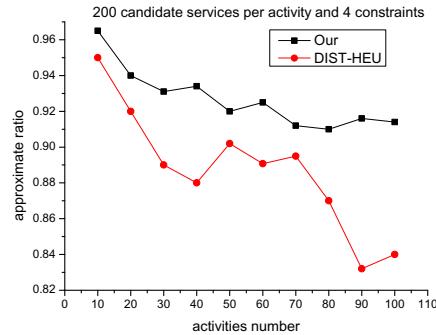


Fig. 3. Varying the number of activities

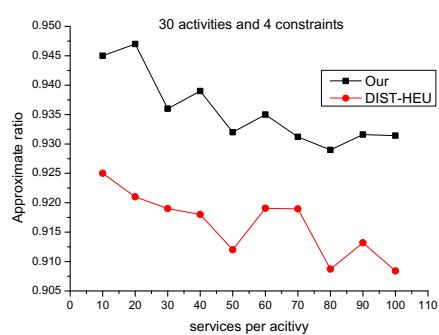


Fig. 4. Varying the number of candidate services per activity

5 Conclusion

In the paper, we propose a service selection based on resource-directive decomposition and the experimental results show that it can obtain a better tradeoff between the computational complexity and the quality of candidate services. In future work, we will use the real dataset such as QWS dataset v2 [1] to test our approach and plan to incorporate with trust model [8].

Acknowledgments. This research was supported in part by the Major International Collaborative Research Project of National Natural Science Foundation of China (NO: 61020106009).

References

1. Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the world wide web. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, pp. 795–804. ACM, New York (2008)

2. Alrifai, M., Risse, T., Dolog, P., Nejdl, W.: A scalable approach for QoS-based web service selection. In: Feuerlicht, G., Lamersdorf, W. (eds.) ICSOC 2008. LNCS, vol. 5472, pp. 190–199. Springer, Heidelberg (2009)
3. Alrifai, M., Risse, T., Nejdl, W.: A hybrid approach for efficient web service composition with end-to-end qos constraints. ACM Trans. Web 6(2), 7:1–7:31 (2012)
4. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Trans. Softw. Eng. 33(6), 369–384 (2007)
5. Bayen, A.M., Sun, D., Clinet, A.: A dual decomposition method for sector capacity constrained traffic flow optimization. Transportation Research 45, 880–902 (2011)
6. Fisher, M.L.: The lagrangian relaxation method for solving integer programming problems. Management Science 50(12), 1861–1871 (2004)
7. Geoffrion, A.M.: Primal resource-directive approaches for optimizing nonlinear decomposable systems. Operations Research 18(3), 375–403 (1970)
8. Jiang, D., Xue, J., Xie, W.: A reputation model based on hierarchical bayesian estimation for web services. In: Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design, pp. 88–93 (2012)
9. Sun, S.X., Zhao, J.: A decomposition-based approach for service composition with global qos guarantees. Inf. Sci. 199, 138–153 (2012)
10. Wang, S.-H.: An improved stepsize of the subgradient algorithm for solving the lagrangian relaxation problem. Computers & Electrical Engineering 29(1), 245–249 (2003)
11. Yu, Q., Bouguettaya, A.: Multi-attribute optimization in service selection. World Wide Web 15(1), 1–31 (2012)
12. Yu, T., Lin, K.-J.: Service selection algorithms for web services with end-to-end qos constraints. In: Proceedings of the IEEE International Conference on e-Commerce Technology, CEC 2004, pp. 129–136 (July 2004)
13. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: Proceedings of the 12th International Conference on World Wide Web, WWW 2003, pp. 411–421. ACM, New York (2003)
14. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. 30(5), 311–327 (2004)

Collaborative Filtering Based on Rating Psychology

Haijun Zhang¹, Chunyang Liu², Zhoujun Li¹, and Xiaoming Zhang¹

¹ School of Computer Science and Engineering, Beihang University, Beijing, China, 100191

² National Computer Network Emergency Response Technical

Team/Coordination Center of China, Beijing 100029

haijun_cumtb@yahoo.com.cn, lcy@isc.org.cn,

{lizj,yolixs}@buaa.edu.cn

Abstract. Nowadays, products are increasingly abundant and diverse, which makes user more fastidious. In fact, user has demands on a product in many aspects. A user is satisfied with a product usually because he or she likes all aspects of the product. Even only few of his or her demands or interests did not be satisfied, the user will have a bad opinion on the product. Usually, user's rating value for an item can be divided into two parts. One is influenced by his or her rating bias and other user's rating for the item. The other is determined by his or her real opinion on the item. The process of rating an item can be considered as an expression of user's psychological behavior. Based on this rating psychology, a novel collaborative filtering algorithm is proposed. In this algorithm, if one latent demand of the user is not satisfied by the item, the corresponding rating value will be multiplied by a penalty value which is less than 1. The parameters in the model are estimated using stochastic gradient descent method. Experiment results show that this algorithm has better performance than state-of-the-art algorithms.

Keywords: Collaborative Filtering, Rating Psychology, Latent Demands, Stochastic Gradient Descent.

1 Introduction

Recommendation system aims at finding favorite items for users. In the past decades, many recommendation algorithms have been proposed, among of which collaborative filtering (CF) attracts much more attention because of its high recommendation accuracy and wide applicability. These CF algorithms can be grouped into two categories, i.e., memory-based CF and model-based CF [1,2]. Memory-based CF mainly includes user-based CF and item-based CF. User-based CF supposes that the test user will like the items which are also liked by the users similar to the test user. Thus, the rating of the test user for the test item is predicted based on the ratings of these similar users. Item-based CF assumes that the test user will like the items which are similar to the items that are liked by the test user in the past. The rating of the test user for the test item is predicted based on the ratings of these similar items. The key step of memory-based CF is the calculation of similarity. Usually, the similarity is calculated based on the user-item rating matrix directly. In this matrix, each row denotes a feature vector

of a user, while each column denotes a feature vector of an item. The similarity between two vectors depends on the elements at which both the two vectors have rating. If the data is very sparse, the number of commonly rated elements of the two vectors is very small, which results in the computed similarity incredible and hence causes low recommendation accuracy. On the other hand, with the growth of users and items, the complexity of similarity computation increases in a non-linear tendency, which restricts its scalability.

To solve these problems, a variety of machine learning and data mining models are proposed, which lead to the appearance of model-based CF algorithms. The basic idea of model-based CF algorithms is trying to find the potential structure of rating matrix to cope with the scalability and data sparsity problems. Model-based CF mainly includes clustering methods [3,4], regression methods [5], graph model based methods, transfer learning method [7], matrix factorization methods, and neural network methods. Graph model based methods include Bayesian network [8,9], PLSA (Probabilistic Latent Semantic Analysis)[10,11], LDA (Latent Dirichlet Allocation)[12], and etc. Matrix factorization methods include SVD (Singular Value Decomposition)[13-17], PCA [18] (Principal Component Analysis), and NMF (Nonnegative Matrix Factorization) [19]. Neural network methods include RBM (Restrained Boltzmann Machine) [20], and etc. Usually, model-based CF has better performance than memory-based CF, but the recommendation made by memory-based CF has good interpretability and is easier to convince users.

However, all these algorithms do not fully consider user's rating psychology. An algorithm based on user's rating psychology can mine the latent structure of rating matrix more effectively. Moreover, it may be more tolerant against data sparsity. Based on this idea, a novel CF algorithm which is more consistent with user's rating psychology is proposed in this paper. The parameters in the model are learned using stochastic gradient descent method. Experiment results show that this algorithm has better performance than state-of-the-art algorithms.

The rest of this paper is organized as follows. Section 2 describes the related work. In section 3, the algorithm based on user's rating psychology is introduced. Some experiments are designed and analyzed in section 4, followed by the conclusions in section 5.

2 Related Work

Collaborative filtering algorithms predict user's interest based on his past feedbacks. User's feedbacks can be explicit or implicit (e.g., purchase of a product, rating on an item, and click on a hyperlink). In rating score prediction field, user's feedbacks are represented by a user-item rating matrix. However, more than 90% of the data is missing in this matrix, i.e., many users only rate a few items. Moreover, there are a lot of noises in the rating data. How to use these sparse data with noises? The method based on matrix low-rank approximation is applied. Low-rank approximation method can retain the main information of the matrix, and meanwhile, it can also remove some noises from the data. SVD is one of low-rank approximation methods, and it is

introduced into CF by Sarwar et al. [13]. However, it needs to compute singular values of the matrix, which limits its scalability. S.Funk [15] suggests learning the latent factor matrices directly by fitting the observed data only, which can be expressed as:

$$\hat{r}_{ui} = p_u^T q_i \quad (1)$$

$$f = \sum_{(u,i) \in Tr} (r_{ui} - p_u^T q_i)^2 + \lambda \left(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 \right) \quad (2)$$

where \hat{r}_{ui} denotes the estimated rating of user u ($u=1, 2, \dots, M$) for item i ($i=1, 2, \dots, N$), and r_{ui} denotes the true rating value of user u for item i , and p_u denotes the latent factors vector of user u , and q_i denotes the latent factors vector of item i . The dimensionality of the latent factors vector is a super parameter which should be set beforehand, and it is usually less than M and N . “For a given item i , the elements of q_i measure the extent to which the item possesses those factors, positive or negative. For a given user u , the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, $p_u^T q_i$ captures the interaction between user u and item i —the user’s overall interest in the item’s characteristics. This approximates user u ’s rating of item i .”— Y. Koren [17]. The parameters p_u and q_i are estimated by minimizing the loss function f which is expressed by Eq. (2), where λ is a regularization constant that is applied to parameters to avoid over-fitting, which is determined by cross validation. Based on this model, Y. Koren [17] added user’s bias and item’s bias, and proposed a modified model which is called RSVD. It can be described as Eq. (3).

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i \quad (3)$$

where μ is the average over all rating values in training set, and b_u is the bias of user u , and b_i is the bias of item i . Here, b_u and b_i measure the rating deviation of user u and item i from overall average respectively, and they need to be estimated using training set. RSVD achieved great success in Netflix Prize competition.

However, both SVD [15] and RSVD [17] suppose that the interaction between a user and an item is the inner product of their corresponding latent factors vectors. It means that the user’s opinion of the item is the summation of the satisfactions that the item brings to the user in different aspects. Actually, a user gives bad evaluation to an item because the item cannot satisfy only few of the user’s demands. If the user does not like the item at all, he or she will not purchase it. For example, the user saw the movie, it means that he or she may like the topic, the actor, or the director of the movie [21]. It means that the rating value of a user for a product does not depend on what he or she got, but mainly depends on what he or she did not get. Based on this idea, this paper proposed a novel CF model. This model differs from the works described above in the following aspects.

1. The demands of user u are denoted by a latent vector p_u . The dimensionality of the vector denotes the number of user's different demands. Each element of p_u measures the quantity of his or her specific demand. The supplies of item i are denoted by a latent vector q_i , and each element of q_i denotes the quantity of its specific supply. The distance between p_u and q_i denotes the gap between user u 's demands and item i 's supplies.
2. The distance between p_u and q_i is a weighted summation of the distance in each dimension. The weight in each dimension is user-specific which reflects that the user gives different importance to his or her different demands.
3. If one of the user's demands cannot be satisfied by the item, the user will give a discount to the rating for the item. In this paper, a negative exponential function is used to denote the discount.

3 Collaborative Filtering Based on Rating Psychology

In this section, user's rating psychology is described and a CF model based on rating psychology is proposed. Then, the parameter learning method is described.

3.1 Rating Psychology

In fact, RSVD takes into account of user's rating psychology to some extent. Under same condition, user u may give bigger rating score to the item than user v does. Thus, RSVD uses rating bias b_u or b_v to represent this individual psychology character. As for items, it uses b_i to capture the bias of item i . Meanwhile, RSVD supposes that the user's opinion of the item is the summation of satisfactions the item brings to the user in each aspect.

However, nowadays, products are increasingly abundant and diverse, which makes user more fastidious, because he or she has many choice. On the other hand, the ads usually advocate the advantages of the product and never mention the shortcomings of the product, which enlarges users' expectations for the product's benefits and reduces users' tolerances to the product's defects. Thus, when a user purchased a product and found its defect, the user usually could not tolerate it even the defect was negligible. Then, he or she would return it or give a bad evaluation to it. On the other hand, if a user does not like the product at all, he or she will not purchase it. So, a user gives high evaluation for a product usually because he or she likes all aspects of the product, and a user gives bad evaluation for a product usually because the product cannot satisfy only few of the user's demands. It means that the rating value of a user for a product does not depend on what he or she got from the product, but mainly depends on what he or she did not get. Moreover, only when the supplies of the product are just equal to user's demands, it can satisfy the user. For example, a user likes horror films, but too scary movies will make him/her uncomfortable.

Based on this rating psychology, a novel model is designed in this paper to mine the latent structure of rating matrix to deal with data sparsity problem.

3.2 Collaborative Filtering Model Based on Rating Psychology

The CF model based on rating psychology is proposed as Eq. (4).

$$\hat{r}_{ui} = \mu + b_u + b_i + \beta_u e^{-\sum_{k=1}^K h_{uk} \cdot (p_{uk} - q_{ik})^2} \quad (4)$$

where $\beta_u \geq 0$ and $h_{uk} \geq 0$. Here, β_u denotes the score related to user u 's demands. This equation indicates that the rating score of user u for item i can be divided into two parts. One is related to user u 's demands, and it is denoted by β_u . The other is not related to his demand but related to rating bias. In extreme situations, if a user v 's opinion on the items is completely under the influence of other user's opinion and is not relevant to his or her true feeling about the items, his β_v is zero. K denotes the number of user's latent demands, i.e., the dimensionality of user's demand, which is a super parameter and should be set beforehand. Parameter p_{uk} denotes the quantity of user u 's k -th demand, and q_{ik} denotes the quantity of item i 's supply for user's k -th demand. Parameters p_{uk} and q_{ik} can be positive or negative, which does not affect their meanings endowed here, because if all of them are added with an enough big positive number, they will be positive and it does not change the estimated rating. In Eq. (4), h_{uk} is the weight of user u in his k -th latent demand. If the k -th demand of user u does not be satisfied by item i , the β_u will be multiplied by $e^{-h_{uk}(p_{uk}-q_{ik})^2}$ as a discount. The parameters b_u , b_i , β_u , p_{uk} , q_{ik} , and h_{uk} need to be estimated from training set.

3.3 Parameters Learning

Stochastic gradient descent method is used to estimate the parameters. First, the loss function is designed as Eq. (5).

$$f = \sum_{(u,i) \in Tr} (r_{ui} - \hat{r}_{ui})^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right) + \lambda_2 \left(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 \right) + \lambda_3 \sum_u \beta_u^2 + \lambda_4 \sum_u \|h_u\|^2 \quad (5)$$

where the first term on the right side of the formula is the sum of square error, and the other four terms on the right side are the regularization terms applied to parameters to avoid over-fitting. In Eq. (5), Tr denotes the training set, and λ_1 , λ_2 , λ_3 , λ_4 are regularization constants which are determined by cross validation. Vectors $p_u = \{p_{u1}, p_{u2}, \dots, p_{uK}\}^T$, and $q_i = \{q_{i1}, q_{i2}, \dots, q_{iK}\}^T$, and $h_u = \{h_{u1}, h_{u2}, \dots, h_{uK}\}^T$, and they denote the latent demands vector of user u , the latent supplies vector of item i , and the weights vector of demands of user u respectively. Then, the gradient of each parameter is calculated by Eq. (6)-(13). For presentation simplicity, notations ψ_{ui} and e_{ui} are introduced.

$$\psi_{ui} = e^{-\sum_k h_{uk} \cdot (p_{uk} - q_{ik})^2} \quad (6)$$

$$e_{ui} = r_{ui} - \mu - b_u - b_i - \beta_u \cdot \psi_{ui} \quad (7)$$

$$\frac{\partial f}{\partial b_u} = -\sum_i 2e_{ui} + 2\lambda_1 b_u \quad (8)$$

$$\frac{\partial f}{\partial b_i} = -\sum_u 2e_{ui} + 2\lambda_1 b_i \quad (9)$$

$$\frac{\partial f}{\partial p_{uk}} = \sum_i [2e_{ui} \cdot \beta_u \cdot \psi_{ui} \cdot h_{uk} \cdot (p_{uk} - q_{ik})] + 2\lambda_2 p_{uk} \quad (10)$$

$$\frac{\partial f}{\partial q_{ik}} = \sum_u [-2e_{ui} \cdot \beta_u \cdot \psi_{ui} \cdot h_{uk} \cdot (p_{uk} - q_{ik})] + 2\lambda_2 q_{ik} \quad (11)$$

$$\frac{\partial f}{\partial \beta_u} = -\sum_i 2e_{ui} \cdot \psi_{ui} + 2\lambda_3 \beta_u \quad (12)$$

$$\frac{\partial f}{\partial h_{uk}} = \sum_i 2e_{ui} \cdot \beta_u \cdot \psi_{ui} \cdot (p_{uk} - q_{ik})^2 + 2\lambda_4 h_{uk} \quad (13)$$

After that, every rating r_{ui} in training set Tr is used to update the parameters by stochastic gradient descent method [16]. The update rules are expressed by the formulas (14)-(19), which can be deduced from the formulas (6)-(13).

$$b_u = b_u + \gamma_1 (e_{ui} - \lambda_1 b_u) \quad (14)$$

$$b_i = b_i + \gamma_1 (e_{ui} - \lambda_1 b_i) \quad (15)$$

$$p_{uk} = p_{uk} - \gamma_2 (e_{ui} \cdot \beta_u \cdot \psi_{ui} \cdot h_{uk} \cdot (p_{uk} - q_{ik}) + \lambda_2 p_{uk}) \quad (16)$$

$$q_{ik} = q_{ik} + \gamma_2 (e_{ui} \cdot \beta_u \cdot \psi_{ui} \cdot h_{uk} \cdot (p_{uk} - q_{ik}) - \lambda_2 q_{ik}) \quad (17)$$

$$\beta_u = \max(0, \beta_u + \gamma_3 (e_{ui} \cdot \psi_{ui} - \lambda_3 \beta_u)) \quad (18)$$

$$h_{uk} = \max(0, h_{uk} - \gamma_4 (e_{ui} \cdot \beta_u \cdot \psi_{ui} \cdot (p_{uk} - q_{ik})^2 + \lambda_4 h_{uk})) \quad (19)$$

For each rating r_{ui} in training set Tr , ψ_{ui} and e_{ui} are computed first according Eq. (6) and (7). Then, all the parameters b_u , b_i , β_u , p_{uk} (for each $k=1, 2, \dots, K$), q_{ik} (for each $k=1, 2, \dots, K$), and h_{uk} (for each $k=1, 2, \dots, K$) are updated. Learning ratio parameters $\gamma_1, \gamma_2, \gamma_3$ and γ_4 are determined by cross validation.

3.4 Workflow of the Algorithm

Finally, the pseudo codes of our algorithm are shown in Figure 1, where we update the parameters using the observed rating values. For description simplicity, this algorithm is called CFBRP (Collaborative Filtering based on Rating Psychology). In the two experiments described in section 4, the initial values of the estimated parameters are set as follows. For each u , i , and k , we set $b_u = b_i = 0$, $\beta_u = 3$, $h_{uk} = 1$. Parameters p_{uk} , and q_{ik} are generated according to two steps as follows: Step 1, sample p'_{uk} randomly from uniform distribution on $[0, 1]$; Step 2, normalize them, i.e., set $p_{uk} = p'_{uk} / \sum_k p'_{uk}$. It is the same way to initialize q_{ik} .

After the parameters are obtained by the algorithm, the ratings on the test set can be estimated by Eq. (4).

-
1. Input: training set Tr
 2. Output: $b_u, b_i, p_u, q_i, \beta_u, h_u$
 3. Initialize parameters $b_u, b_i, p_u, q_i, \beta_u, h_u, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2, \lambda_3$, and λ_4 .
 4. For each iteration
 5. Randomly permute the data in Tr .
 6. For each r_{ui} in Tr
 7. Step 1. Compute ψ_{ui} by Eq.(6);
 8. Step 2. Compute e_{ui} by Eq.(7);
 9. Step 3. Update the parameters as shown in Eq. (14)-(19).
 10. End for
 11. Decrease the learning ratios $\gamma_1, \gamma_2, \gamma_3, \gamma_4$; they are multiplied by 0.9.
 12. End for
-

Fig. 1. The Algorithm of Collaborative Filtering based on Rating Psychology (CFBRP)

4 Experiments

In this section, two experiments are designed to evaluate the algorithm CFBRP. One is to compare it with RSVD on MovieLens¹ 100k data set. RSVD is a well-known algorithm which has high prediction accuracy on the benchmark data sets. The other experiment is designed to compare CFBRP with other state-of-the-art algorithms reported in [22] to test its performance against data sparsity.

¹ <http://www.grouplens.org/>

4.1 Experiment Designed

To compare CFBRP with the algorithm RSVD, MovieLens 100k data set is used to evaluate the algorithms. MovieLens 100k data set has 100000 rating scores rated by 943 users on 1682 items (1-5 scales), where each user has more than 20 ratings. We randomly split MovieLens 100k data set into 5 parts which are used for 5-fold cross validation. This revised data set is called data set 1.

We use MovieLens 10M data set to compare CFBRP with other state-of-the-art algorithms reported in [22]. This data set contains 10000054 rating scores rated by 71567 users on 10681 movies. It is randomly split into 5 parts which are used for 5-fold cross validation. For each training set, 80%, 50%, 30%, 20% and 10% of the data are randomly selected to represent 5-level data sparsity degree, and they constitute 5 observed data sets respectively. Then, these observed data sets are used to train the algorithm respectively, and the corresponding test data set is used to evaluate the algorithm. At each sparsity level and on each test data set, run the algorithm CFBRP 5 times independently, and we report the average values in this paper. This revised data set is called data set 2.

In order to compare the prediction quality of our method with other methods, we use MAE (mean absolute error) as evaluation metric. The MAE is calculated by averaging the absolute deviation between predicted values and true values:

$$MAE = \frac{\sum_{(u,i) \in Te} |r_{ui} - \hat{r}_{ui}|}{|Te|} \quad (20)$$

where $|Te|$ is the number of tested ratings in the test set Te . The lower the MAE is, the better the performance is.

4.2 Comparison with Algorithm RSVD

To compare algorithm CFBRP with algorithm RSVD, both of them are evaluated by a 5-fold cross validation on data set 1 at different dimensionality (from 10 to 100 at interval of 10) respectively. For CFBRP, the dimensionality means the number of user's latent demands or interests, and for RSVD, it means the number of latent factors. Both the algorithms are carried out 5 times independently at each test set, which means that at each dimensionality each algorithm is run 25 times, and we report the average values. The initial values of parameters in RSVD are set similarly to that of Koren's paper [16]. The initial values of learning ratio parameters and regularization constants in CFBRP are set as follows: $\gamma_1 = 0.015$, $\gamma_2 = 0.5$, $\gamma_3 = 0.01$, $\gamma_4 = 0.1$, $\lambda_1 = 0.1$, $\lambda_2 = 0.001$, $\lambda_3 = 0.015$, and $\lambda_4 = 0.001$. The comparison results are shown in Fig.2.

From figure 2, it is shown that with the increasing of the dimensionality the MAE of both algorithms has a decrease tendency, and the performance of CFBRP is better than RSVD when the dimensionality is bigger than 30. When the dimensionality is set to 100, the MAE of CFBRP is 0.7137 while the MAE of RSVD is 0.7163. Algorithm CFBRP can get a better performance than RSVD, because CFBRP exploits user's rating psychology more carefully and reasonably as described in section 2 and section 3.

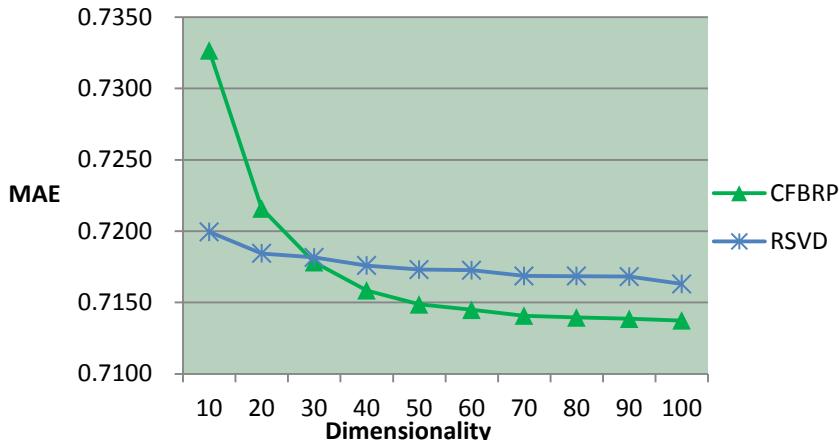


Fig. 2. Effect of Dimensionality on MAE

4.3 Comparison with Other State-of-the-Art Algorithms

To evaluate the tolerance of CFBRP against data sparsity, we compare CFBRP with other state-of-the-art algorithms reported in [22] on data set 2. These algorithms include SVD [15], PMF [23], and TagRec [22]. For compare them consistently, the dimensionality K in CFBRP is set to 20, as done in [22]. The initial values of learning ratio parameters and regularization constants in CFBRP are set as follows: $\gamma_1 = 0.015$, $\gamma_2 = 0.5$, $\gamma_3 = 0.01$, $\gamma_4 = 0.1$, $\lambda_1 = 0.01$, $\lambda_2 = 0.001$, $\lambda_3 = 0.015$, and $\lambda_4 = 0.001$. The reported MAE of CFBRP in table 1 is the average value over 25 independent runs.

Table 1. Comparison on MAE with the Algorithms Reported in [22]

Training Data	SVD	PMF	TagRec	CFBRP
80%	0.6167	0.6156	0.6145	0.6168
50%	0.6349	0.6337	0.6307	0.6295
30%	0.6570	0.6569	0.6494	0.6470
20%	0.6776	0.6766	0.6650	0.6637
10%	0.7264	0.7089	0.6962	0.6836

Table 1 shows that, when using 80% of the training data, CFBRP cannot do better than the other algorithms, and the reason may lie in the too small dimensionality. Figure 2 also shows that when the dimensionality is set to a small value, the performance of CFBRP is less than RSVD. However, when less than 50% of the training data is used, CFBRP has better performance than all other algorithms. It implies that, by exploiting the rating psychology of users, CFBRP has better tolerance against data sparsity even the algorithm TagRec utilizes other information, i.e., tag information.

It hence means that, rating psychology is very important information implied in rating matrix, and it can be used to improve the performance of collaborative filtering.

5 Conclusions

The process of rating an item can be considered as an expression of user's psychological behavior. How to use user's psychology characteristics to improve recommendation is a meaningful research field. In this paper, a novel recommendation algorithm is proposed by utilizing the rating psychology. Experiment results show that the proposed algorithm has a better performance than other competitors. But, how to prove the effectiveness of the algorithm theoretically is a problem we will study in the future. In future works, we will also explore other useful psychology information to improve the performance of recommendation algorithms. Some optimization processes are also need to be studied, such as how to speed up the algorithm, and how to deploy it in a distributed computing environment to cope with big data.

Acknowledgement. This work was supported by the National Natural Science Foundation of China [grant number 60973105, 90718017, 61170189, and 61202239], the Research Fund for the Doctoral Program of Higher Education [grant number 20111102130003], and the Fund of the State Key Laboratory of Software Development Environment [grant number KLSDE-2011ZX-03].

References

1. Adomavicius, G., Tuzhilin, A.: Towards the Next Generation of Re-commander Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 634–749 (2005)
2. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. In: *Advances in Artificial Intelligence*, pp. 1–19 (2009)
3. O'Connor, M., Herlocker, J.: Clustering items for collaborative filtering. In: *Proceedings of the ACM SIGIR Workshop on Recommender Systems, SIGIR 1999* (1999)
4. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In: *Proceedings of the Fifth International Conference on Computer and Information Technology* (2002)
5. Canny, J.: Collaborative filtering with privacy via factor analysis. In: *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2002)
6. Vucetic, S., Obradovic, Z.: Collaborative Filtering Using a Regression-Based Approach. *Knowledge and Information Systems* 7(1), 1–22 (2005)
7. Li, B., Yang, Q., Xue, X.: Can movies and books collaborate?—Cross-domain collaborative filtering for sparsity reduction. In: *Proceedings of IJCAI 2009*, pp. 2052–2057 (2009)
8. Su, X., Khoshgoftaar, T.M.: Collaborative Filtering for Multi-class Data Using Belief Nets Algorithms. In: *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence* (2006)

9. Miyahara, K., Pazzani, M.J.: Collaborative filtering with the simple Bayesian classifier. In: Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence (2000)
10. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Transactions on Information Systems 22(1), 89–115 (2004)
11. Hofmann, T., Puzicha, J.: Latent class models for collaborative filtering. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 688–693 (1999)
12. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. Journal of Machine Learning Research 3(4-5), 993–1022 (2003)
13. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Application of Dimensionality Reduction in Recommender System-A Case Study. In: ACM 2000 KDD Workshop on Web Mining for e-commerce-Challenges and Opportunities (2000)
14. Sarwar, B., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Fifth International Conference on Computer and Information Science (2002)
15. Funk, S.: Netflix update: Try this at home, Tech. Rep. (2006), <http://www.sifter.org/~simon/journal/20061211.html>
16. Koren, Y.: Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In: Proceedings of KDD 2008 (2008)
17. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (2009)
18. Goldberg, K., Roeder, T., Gupta, D., et al.: Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval 4(2), 133–151 (2001)
19. Chen, G., Wang, F., Zhang, C.: Collaborative Filtering Using Orthogonal Nonnegative Matrix Tri-factorization. Information Processing & Management 45(3), 368–379 (2009)
20. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of 24th Annual International Conference on Machine Learning (2007)
21. <http://x1vector.net/blog/?p=179>
22. Zhou, T.C., Ma, H., King, I., et al.: Tagrec: Leveraging tagging wisdom for recommendation. In: Proceedings of the 2009 International Conference on Computational Science and Engineering, vol. 4, pp. 194–199 (2009)
23. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, vol. 20 (2008)

Modeling Semantic and Behavioral Relations for Query Suggestion

Jimeng Chen^{1,*}, Yuan Wang¹, Jie Liu¹, and Yalou Huang^{1,2}

¹ College of Information Technical Science, Nankai University, Tianjin, 300071

² College of Software, Nankai University, Tianjin, 300071

{jeanchen,yayaniuzi23}@mail.nankai.edu.cn,

{jliu,huangy1}@nankai.edu.cn

Abstract. Query suggestion helps users to precisely express their search intents. The state-of-the-art approaches make great progress on high-frequency queries via click graphs. However, due to query ambiguity and click-through data sparseness, these approaches are limited to reality applications. To address the issue, this paper models both semantic and behavioral relations on hybrid bipartite graphs from click logs. Firstly, to overcome the sparseness, a semantic relation graph is established by multiple morphemes (queries, keywords, phrases and entities), independently of clicks. And then semantic relations between queries and other morphemes on the graph are used to find similar queries for query description. Secondly, in order to find related queries for multiple user intents, a behavioral relation graph is constructed by three kinds of user behaviors. Global clicks between queries and URLs display multiple intents by all users; local clicks imply personal preference by a single user; and query formulations in a session represent relations between queries. Finally, two hybrid methods are proposed to combine both semantic and behavioral relations to suggest related queries. To illustrate our methods, we employ the AOL query log data for query suggestion tasks. Experimental results demonstrate that more than 46.5% of queries get improved suggestions compared with the baseline click models.

Keywords: query suggestion, semantic and behavioral relations, hybrid graphs, click-through data, user preference.

1 Introduction

Users submit queries to express their information needs in a typical search scenario. However, most of time, original queries can not accurately express users' search intention. On one hand, queries are too short to make intentions clear. By analysis of the AOL click logs recording over three months in 2006, we observe that 14% of queries contain a single word, and 78.6% contain less than 4 words. On the other hand, queries are always ambiguity or various intentions[1]. For example, "apple" may refer to "fruit", "a company" or "electronic products". In addition, sometimes users are lack of enough knowledge to construct a succinct and precise query.

* Corresponding author.

In order to help users describe their exact information needs, search engines widely employ query suggestion technologies. Typically, query suggestion is based on document analysis[2], or query log analysis[3,4]. The former techniques mainly find similar terms in related documents, while the latter mainly find queries that have similar click behavior. Because of sparseness of clicks in logs, fuse models[5] combining clicks information and relevant documents are also studied. As important guiding information from people, click graphs in a query log have been widely studied to mine user preference and user behavior for query suggestion. We define the task: Given a query and a query log, the system recommends a list of semantic similar or related queries to the given query.

In this paper, we study both semantic relations and user behavioral relations from click-through data to suggest proper queries for multiple user intentions. At the beginning, query semantics and user behaviors in a log are analyzed separately. Then, two hybrid methods considering both sides are proposed. The details are as follows: On one hand, in order to avoid the sparseness of click-through data, we establish a semantic relation graph ("semantic graph" for short) to describe a query. The graph is a hybrid bipartite graph, in which nodes are morphemes such as queries, keywords, phrases and entities, and edges are semantic relations between queries and other types of nodes. Hence, similar queries are collected as descriptions for the source query by searching corresponding morphemes on the semantic graph. On the other hand, by analyzing different user behaviors, we construct a behavioral relation graph ("behavioral graph" for short) to find related queries for multiple user intents and preferences. It is a hybrid bipartite graph built by multiple granularity. A coarse-grained graph is built by global clicks of all users to describe the common behaviors, and a fine-grained one is built by a local click history of a single user to describe personal preference, and a third one is by temporary relationship between queries. As a result, three bipartite graphs are established: a global query-URL bipartite graph, a local user-query bipartite graph and a query-query bipartite graph extracted from sessions. Furthermore, two hybrid suggestion methods are proposed to fuse both semantic and user behavioral relations in the log. We consider unsupervised linear combinations, and supervised machine learning combinations respectively. Experimental results show that the hybrid methods generate semantic related queries to users, and improve the recommendation quality as well.

The rest of the paper is organized as follows. We discuss related work in the next section. Section 3 introduces common information for click graphs. We propose two hybrid methods with semantic and behavioral graphs in Section 4. Section 5 presents experiments and the empirical analysis of our models. Finally, conclusions are given in Section 6.

2 Related Work

Query suggestion based on log analysis attracts considerable attention in recent years. Click logs provide researchers a whole view of users' behavior during search, including submitting queries, clicking URLs and formulating queries. According to different user behaviors, methods can mainly be divided into three categories: click-based, session-based and content-based methods. In order to find related queries, click-based

approaches mainly discover global relationship between queries and URLs by all users, while session-based ones provide a detailed analysis to individuals' activities during search. Content-based methods focus on finding similar queries that have similar semantic relevance.

The main idea of click-based methods is that if people click similar URLs for two queries, then the two are related. Many research works[3,4,6] employ random walk algorithms on a click bipartite graph to suggest related queries. Specially, Deng etc[4] propose a CF-IQF model considering that URLs have different importance for each query. Because of its outstanding performance, we employ this model as our basic model. Ma etc[7] construct both query-URL and user-query bipartite graphs to find related queries, then a content similarity metric is applied to find semantic related queries. It is a similar work as ours, but we consider both semantic and behavioral relations by establishing a semantic graph and a behavioral graph instead.

In session-based approaches, they share the assumption that users express the same intent in a short period of time, so that queries are related in the same session. Hence, similar metric[8], Mutual Information using co-occurrence[9], time distance between consecutive queries[10], and sequence analysis[11] are applied to query suggestion. Specially, [10] considers that queries are more relevant when they are closer by time in a session. We build a bipartite graph between queries in sessions, so that queries in the same session still keep association with each other.

In content-based methods, researchers mainly find similar queries to suggest. Because queries are consisted of terms and marks as texts, traditional methods on text mining are able to utilize. In order to describe a query, Bendersky and Croft[12] build a hypergraph using various morphemes such as terms, bigrams, nouns and entities. However, a query is too short to find relatedness with each other. To overcome the problems, researchers do query expansion using relevant documents. Sahami etc[13] select top K pseudo-relevance documents returned by a search engine to describe a query. Thus, query similarity can be replaced by document similarity.

Moreover, some fuse methods are proposed. Yates[14] identifies five relation graphs from a query log. In all graphs, nodes are queries, and edges are introduced if two nodes contain common words, common urls, common terms in urls, belong to the same session, have a link between two clicked urls. Kelly etc[15] suggest related queries using term relevance feedback techniques. Yi and Allan[5] add missing clicks to click graphs employing content similarity.

To suggest proper queries for multiple intents, we propose a fuse method combining content-based, click-based and session-based methods. In the next section, we briefly introduce some common information for click graphs.

3 CF-IQF Model

The CF-IQF (Click Frequency Inverse Query Frequency) model[4] is proposed by Deng etc in 2009. It is a query representation model on the click graphs. The main idea is that if two queries are similar, then their corresponding URLs are similar. Furthermore, it employs TF-IDF (Term Frequency Inverse Document Frequency) ideas to limit the high frequency queries. The URLs connecting fewer queries are more important because of

its concentrate. The model gains a good performance, and it can be extended and used for other bipartite graphs. Hence, we employ this model as our basic model.

At the beginning, we introduce the common information of click-through data. A common click-through data contains the following information: a user (u), a query (q) submitted by the user, a URL (d) which the user clicked, and the time (t) when the query was submitted for search. The relatedness between queries in the CF-IQF model is defined as follows,

$$p(q_j|q_i) = \sum_{d_k \in D} p(d_k|q_i)p(q_j|d_k), \quad (1)$$

where $p(d_k|q_i)$ and $p(q_j|d_k)$ denote the transition probability from the query q_i to URL d_k and d_k to q_j respectively. D is the URL set in a log. Formula 1 can be propagated by a random walk between queries and URLs.

Define c_{ij} as click frequency between query q_i and URL d_j , the notations in Formula 1 are defined as

$$p(d_j|q_i) = \frac{cfiqf(q_i, d_j)}{cfiqf(q_i)}, \quad p(q_i|d_j) = \frac{c_{ij}}{cf(d_j)}, \quad (2)$$

where $cfiqf(q_i, d_j) = c_{ij} \cdot iqf(d_j)$, and $cfiqf(q_i) = \sum_{d_j \in D} cfifqf(q_i, d_j)$, and $cf(d_j) = \sum_{q_i \in Q} c_{ij}$. The inverse query frequency for the URL d_j is defined as

$$iqf(d_j) = \log \frac{|Q|}{n(d_j)}, \quad (3)$$

where $|Q|$ is the total number of queries in the query log, and $n(d_j)$ is the frequency of query which clicks URL d_j .

4 Query Suggestion by Semantic and Behavioral Relations

In this section, we propose a fuse framework to suggest queries by considering semantic and behavioral relations. In order to overcome the sparseness of click-through data, we build a semantic relation graph to describe a query by content-based methods. To find related queries for different user intents and preferences, we construct behavioral relation graphs by clicks and queries in sessions. Finally, two hybrid methods are proposed to suggest semantic related queries for multiple intents.

4.1 Finding Similar Queries by a Semantic Relation Graph

In this section, we detail how to construct a semantic graph, and then semantic relations are used to find similar queries. We consider multiple morphemes of queries to build a semantic graph. As a result, keywords, phrases and entities are extracted from the source queries. Actually, keywords in queries describe the detailed ideas of user intents. However, one keyword can only tell a fragment meaning. Luckily, previous research works[2,16] show that noun phrases have proven to be reliable for key concept discovery in queries. Therefore, phrases that tell more precise meanings are extracted, too. So

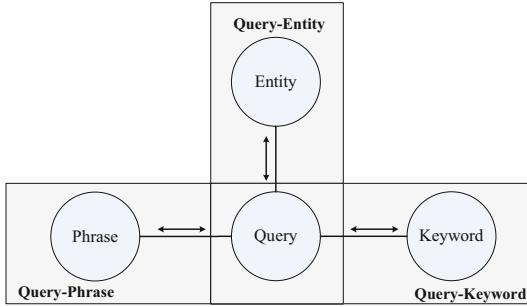


Fig. 1. The semantic graph consists of query-keyword, query-phrase, query-entity bipartite graphs

are entities that stand for the principal components of queries. Furthermore, we construct the semantic graph by combining query-keyword, query-phrase, and query-entity bipartite graphs together, in which edges are semantic relations connecting queries with other types of nodes, as Figure 1 shows. In fact, it is a hybrid graph established by three bipartite graphs. For example, a query such as "mother of the bride dresses" is associated with keywords "mother", "bride" and "dress", associated with phrases "mother" and "the bride", associated with entities "mother", "the bride", "mother of the bride".

Define queries as $Q = \{q_1, q_2, \dots, q_n\}$ and keywords as $K = \{k_1, k_2, \dots, k_m\}$. A query-keyword bipartite graph is an undirected graph, defined as $B_{qk} = (V_{qk}, E_{qk})$ where nodes are $V_{qk} = Q \cup K$, and edges are $E_{qk} = \{(q_i, k_j)\}$. The edge e_{ij} is weighted based on frequency of k_j appearing in q_i . Similar definitions are defined for query-phrase and query-entity bipartite graphs, denoted as B_{qp} and B_{qe} respectively.

According to the main idea that similar queries would be described by similar keywords, phrases and entities, our semantic relation $p_{qs}(q_j|q_i)$ between queries is a combination among three bipartite graphs,

$$p_{qs}(q_j|q_i) = a \cdot p_{qk}(q_j|q_i) + b \cdot p_{qp}(q_j|q_i) + c \cdot p_{qe}(q_j|q_i) \quad (4)$$

where $p_{qk}(q_j|q_i)$, $p_{qp}(q_j|q_i)$ and $p_{qe}(q_j|q_i)$ are relation scores calculated by the CF-IQF model on B_{qk} , B_{qp} , and B_{qe} respectively. Parameters a , b and c are factors of influence of each graph (default set to 1/3), constrained to $a + b + c = 1$.

In details, we introduce how we extract keywords, phrases and entities in query logs. Keywords are terms except stop-words in queries. Phrases are extracted by an English phrase chunk tool named CRFChunk[17]. In our opinion, entities are names, important notions or concepts that user would search separately. So we develop a novel way to extract them. At first, entity candidates are generated by different term permutation of a query. Then a candidate is selected when the candidate submit as a query in the log.

4.2 Finding Related Queries by a Behavioral Relation Graph

In order to identify multiple intents, different user behaviors are mined to construct a behavioral relation graph. To notice both the actives of people and the behavior of a single person, we divide user behaviors into three kinds: In the first category, we study

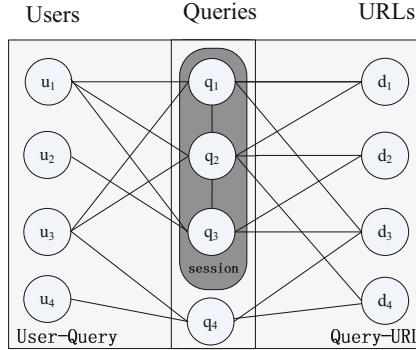


Fig. 2. The behavioral graph consists of query-URL, query-user, and query-query bipartite graphs

mass behavior via clicks between queries and URLs by all users because users always have common activities which display multiple intents; In the second category, personal behavior representing personal preference is studied by an individual search history; In the third category, we analyze query formulation in a session. Therefore, the behavioral graph is established by three bipartite graphs: *a global query-URL bipartite graph* by all users, *a local user-query bipartite graph* and *a local query-query bipartite graph* extracted from sessions.

The global query-URL bipartite graph is established between queries and URLs, in which edges are users' clicks from queries to corresponding URLs. The edge is weighted based on click frequency. Thus it is defined as $B_{qd} = (V_{qd}, E_{qd})$ where $V_{qd} = Q \cup D$, URLs are $D = \{d_1, d_2, \dots, d_l\}$ and $E_{qd} = \{(q_i, d_j)\}$. It is shown as Figure 2.

The local user-query bipartite graph describes relations between users and queries, in which edges are submissions from users to queries. The weights of the edges are frequency of submissions. It is defined as $B_{qu} = (V_{qu}, E_{qu})$ where $V_{qu} = Q \cup U$, users are $U = \{u_1, u_2, \dots, u_r\}$ and $E_{qu} = \{(q_i, u_j)\}$ as Figure 2.

The local query-query bipartite graph represents relatedness between queries in the same sessions, in which weights of edges are similarity by considering both content (such as cosine similarity) and time distance[10]. It is defined as $B_{qq} = (V_{qq}, E_{qq})$ where $V_{qq} = Q \cup Q$ and $E_{qq} = \{(q_i, q_j)\}$. The weights of edges are defined as :

$$f_{session}(q_i, q_j) = \lambda \cdot \cos(q_i, q_j) + (1 - \lambda) \sum_{q_i, q_j \in session} d^{dist(q_i, q_j)}, \quad (5)$$

where $\cos(q_i, q_j)$ calculates cosine similarity; $q_i, q_j \in session$ means q_i and q_j are in the same session; $\sum_{q_i, q_j \in session} d^{dist(q_i, q_j)}$ is time distance summed by every session; $dist(q_i, q_j)$ counts number of queries between q_i and q_j in a session, and d is a constant(varying from 0 to 1, default set to 0.9). λ is a factor of influence of each graph(0.5 as a default).

The global query-URL bipartite graph can straightly employ the CF-IQF model to calculate behavioral relation between queries, defined as $p_{qd}(q_j|q_i)$. As for two local bipartite graphs, we adopt users and queries instead of URLs respectively. Local behavioral relations are denoted as $p_{qu}(q_j|q_i)$ and $p_{qq}(q_j|q_i)$ separately.

Since behavioral relations are affected by user behaviors, our behavioral relation $p_{qb}(q_j|q_i)$ between queries is a combination among three bipartite graphs,

$$p_{qb}(q_j|q_i) = \alpha \cdot p_{qd}(q_j|q_i) + \beta \cdot p_{qu}(q_j|q_i) + \gamma \cdot p_{qq}(q_j|q_i) \quad (6)$$

where $p_{qd}(q_j|q_i)$, $p_{qu}(q_j|q_i)$ and $p_{qq}(q_j|q_i)$ are relation scores calculated by the CF-IQF model on B_{qd} , B_{qu} , and B_{qq} respectively. Parameters α , β and γ are factors of influence of each graph (default set to 1/3), constrained to $\alpha + \beta + \gamma = 1$.

4.3 Associations of Semantic and Behavioral Relations

In order to take advantage of the semantic and behavioral relations, we fuse the two kinds of graphs together. We propose two hybrid methods. One is unsupervised linear combination; the other is supervised machine learning combination to learn importance of the two relations. Notice that, parallel computing can be applied to compute the two relations. Therefore, we do not consider efficiency problems in this paper.

1. Linear Combination of Multiple Graphs

We apply a linear combination between the semantic and behavioral relations via two graphs as follows,

$$R_{lc}(q_j|q_i) = \mu \cdot p_{qb}(q_j|q_i) + (1 - \mu) \cdot p_{qs}(q_j|q_i) \quad (7)$$

where μ is a factor of influence of the behavioral graph.

The list of recommendation is ranked by their relation scores $R_{lc}(q_j|q_i)$. Mark this method as LC-MG (Linear Combination of Multiple Graphs).

2. Machine Learning Combination of Multiple Graph

In this method, we treat query suggestion as a classification task. Positive examples are pairs of relevant queries. We extract six relation scores between two queries as features: $p_{qd}(q_j|q_i)$, $p_{qu}(q_j|q_i)$, $p_{qq}(q_j|q_i)$ from behavioral relations, and $p_{qe}(q_j|q_i)$, $p_{qk}(q_j|q_i)$ and $p_{qp}(q_j|q_i)$ from semantic relations. Next, we employ existed classification models to predict a related query. Finally the list of recommendation is ranked by their prediction scores. Notice that the objective function is according to the selected machine learning algorithm. Mark this method as MLC-MG (Machine Learning Combination of Multiple Graphs).

5 Experimental Results

In the following experiments we compare our proposed models with other methods on the task of query suggestion. In the rest of this section, we introduce the data collection, the evaluation metrics and the results.

Table 1. Details of click log collection after cleaning

month	#session	#query	#avgQuery	#userID
3	382270	1580411	4.13	126955
4	261341	1174407	4.49	93007
5	302657	1254896	4.15	107916
avg	315423	1336571	4.26	109293

5.1 Data Collection

The click dataset that we study is extracted from the query log of AOL search engine[18] over 3 months (from March to May 2006). Each record of the click contains the information: UserID, Query, Time, Rank and ClickURL. To do research on user behavior, we separate the raw data by sessions. In each session, queries are submitted by a unique user, and any two sessions should be more than 30 minutes intervals[19]. As a result, the dataset exists many sessions containing only one click. Since these records have little help to understand users' behavior for query reformulation, we clean the data by removing the sessions that contain less than 2 clicks. Then we combine the near-duplicated queries which have the same terms without the stopwords and punctuation marks (for example, "google.com" and "google" will be combined as the same query). After cleaning, we get totally 946,268 sessions (37.9% of all). Averagely, we collected 315,423 sessions, 1,336,571 queries, 109,293 users per month, and 4.26 queries for each session, see details in Table 1.

Then our collection is divided into two sets, a history set containing the first two months' log and a current set containing the third month's log. The history set is used to construct semantic and behavioral graphs, and the current set is used to generate training and testing queries.

5.2 Evaluation Metrics

We collect 10 training queries and 1000 corresponding queries for each query. Finally we get 10000 pairs of queries. We randomly select 200 testing queries in which 100 test queries are long tail queries, and the other 100 are high frequency ones. We manually evaluate the relevance by 3 levels as follows. No relation: have no relation with the source query; Weak relation: have weak relation or similar with the source query; Close relation: have close relation or the same as the source query.

Each method can suggest 10 related queries at most for each query. And each suggestion is estimated more than 3 persons.

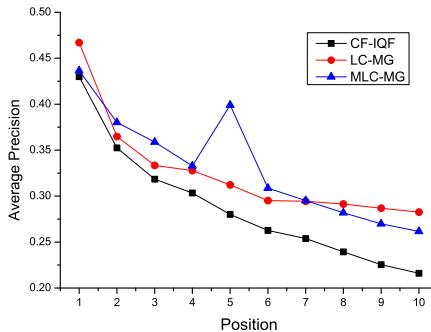
In our experiments, we report the precision from $P@1$ to $P@10$, mean average precision (MAP), the $NDCG@10$ and accuracy. The results are compared to the CF-IQF model by the raw click graph.

5.3 Query Similarity Analysis

In this section, we discuss proposed models (LC-MG and MLC-MG) using combination of semantic and behavioral graphs. Then we make comparisons with the baseline model CF-IQF. The average results on 200 test queries are shown in Table 2.

Table 2. Results of different algorithms

Models	MAP	P@1	P@10	NDCG@10
CF-IQF	40.36%	43.00%	21.60%	43.39%
LC-MG	44.41%	46.70%	28.26%	50.49%
MLC-MG	46.08%	43.65%	26.15%	53.10%
LC-MG /CF-IQF	10.03%	8.61%	30.79%	16.38%
MLC-MG /CF-IQF	14.16%	1.52%	21.06%	22.38%

**Fig. 3.** Performance comparison of different methods

As shown in Table 2, our proposed methods considering both semantic relations and user behavioral relations all boost the baseline method. The MLC-MG using machine learning algorithms get the best predictions especially on *MAP* and *NDCG@10*, increased by 14.16% and 22.38% respectively. And LC-MG using the simple linear combination improves on *P@1* and *P@10*, increased by 8.61% and 30.79% respectively. Figure 3 shows performance of the three methods on precision at different position from 1 to 10. As is shown, the LC-MG gains less precision from *P@2* to *P@7* than MLC-MG. It makes a good explanation that why MLC-MG has better performance at *MAP* and *NDCG*. In addition, the hybrid methods we proposed make a great improvement in query suggestion; more than 46.5% of queries are suggested with proper queries while it doesn't make any recommendation in the baseline model.

We selectively show the top-5 suggestions ranked by the CF-IQF model, MLC-MG model and the Yahoo system in Table 3. In general, all the three models are partially similar. Obviously, Yahoo system mainly suggests similar queries with partial terms the same as the source query, while our model can find multiple intents for queries, for example, "aa" contains two mainly intents:1) "flight" such as "aa.com", "american airlines" and "travel map"; 2)"organization" such as "aa online" and "alcoholic anonymous virginia". Since we consider semantic relations as well as user behaviors, there are three advantages compared with the traditional CF-IQF model. First, we overcome the sparseness of click-through data by semantic relations. A long tail query "mesh motorcycle pants" shares no similar clicks with other queries in the log. So the CF-IQF model is not able to make any recommendation, while the MLC-MG model makes some effective suggestions. Second, queries that have similar terms with the source query will

Table 3. Examples of query suggestion by CF-IQF model, MLC-MG model, and Yahoo system

CF-IQF model	MLC-MG model	Yahoo system
Query = aa		
american airlines alcoholics anonymous aa.com airlines american	aa online aa.com alcoholic anonymous virginia american airlines travel map	aa route planner aa meetings aa jetnet login aa credit union aa meeting locations
Query = travel		
travelocity travel expedia orbitz airline ticket	yahoo travel orbitz travelocity cheap ticket travel channel	travel channel yahoo travel travel insurance expedia travel travel deals
Query = mesh motorcycle pants		
none	motorcycle review motorcycle trailer motorcycle t-shirt motorcycle jacket motorcycle salvage	air mesh motorcycle pants women's mesh motorcycle pants mesh motorcycle pants review ladies mesh motorcycle pants mesh motorcycle pants clearance

take precedence to be suggested. For example, "aa online" for "aa" and "yahoo travel" for "travel". Third, we still maintain user behavioral relation through global query-URL and local user-query and query-query bipartite graphs. Hence, queries that have no semantic relation but have strongly behavioral relation will still be recommended. Related suggestions are generated by MLC-MG such as "alcoholic anonymous virginia" ("aa virginia" for short) for "aa", "orbitz" (a travel website) for "travel" and so on. Therefore, query suggestion benefits from considering both semantic relations and user behavioral relations, and the combination also has advantages in query representation, or other NLP and IR tasks. Since parallel computing can be applied to compute the different relations, we do not consider efficiency problems in this paper.

5.4 Parameters Tuning

In this section, we will introduce details in parameter tuning in experiments. To find out the best combination of LC-MG, we tune parameters (α , β and γ from the behavioral relation in Formula 6; a , b and c from the semantic relation in Formula 4) with values $\alpha : \beta : \gamma = 10^k : 10^k : 10^k$, $k = 0..2$ when $\mu = 0.5$. As a result, we find out that when $\alpha : \beta : \gamma = 100 : 1 : 100$, $a : b : c = 1 : 100 : 1$, the LC-MG method gets best average performance on 10 training queries. It represents that 3 factors play leading roles: global clicked URLs and local reformulated queries from user behavioral relations, and phrases from semantic relations.

With the purpose of finding the best machine learning algorithm for the MLC-MG method, we make use of Weka Toolkit[20]. It is a collection of machine learning algorithms for data mining tasks. We collect 10000 pairs of queries for training. Then, we also

Table 4. Learning results by machine learning methods

Models in Weka	Accuracy
trees.J48	81.06%
trees.LMT	82.30%
trees.RandomForest	84.01%
rules.DecisionTable	78.32%
bayes.NaiveBayes	52.91%
meta.MultiClassClassifier.SMO	51.07%
meta.AttributeSelectedClassifier.RamForesult	83.13%
meta.AdaBoostM1.RandomForest	84.06%
meta.Bagging.RandomForest	84.53%
meta.AttributeSelectedClassifier.RamForesult	83.13%

select 10 popular learning and classification algorithms to train the models, covering rule-based, tree-based, and meta-based methods, processing single classifier and integrated classifiers, as Table 4 shows. Tree-based learning approaches have better accuracy than others among the single classifiers. The Random Forest gets the highest performance in those single classifiers. In general, the integrated classifiers gain better results than the single classifiers. Of all, the bagging method using random forest as classifiers achieves the highest accuracy. Thus, we select bagging as the final learning method.

6 Conclusions

This paper studies the problem of query suggestion using both semantic relations and user behavioral relations in a query log. We propose to use multiple bipartite graphs to mine the relations: semantic relations are mined from keywords, phrases and entities to query bipartite graphs; user behavioral relations are mined from global query-URL bipartite graphs and local user-query, query-query bipartite graphs. In addition, we design two combination approaches, named LC-MG, MLC-MG to seamlessly associate the multiple relations. The experimental results on a real log data demonstrate that the proposed approach outperforms the baseline algorithms with a large margin.

Acknowledgments. This research is supported by the National Natural Science Foundation of China (No. 61105049).

References

1. Song, R., Luo, Z., Nie, J.Y., Yu, Y., Hon, H.W.: Identification of ambiguous queries in web search. *Inf. Process. Manage.* 45(2), 216–229 (2009)
2. Xu, J., Croft, W.B.: Query expansion using local and global document analysis. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1996, pp. 4–11. ACM, New York (1996)

3. Craswell, N., Szummer, M.: Random walks on the click graph. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, pp. 239–246. ACM, New York (2007)
4. Deng, H., King, I., Lyu, M.R.: Entropy-biased models for query representation on the click graph. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, pp. 339–346. ACM, New York (2009)
5. Yi, X., Allan, J.: Discovering missing click-through query language information for web search. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM 2011, pp. 153–162. ACM, New York (2011)
6. Mei, Q., Zhou, D., Church, K.: Query suggestion using hitting time. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, pp. 469–478. ACM, New York (2008)
7. Ma, H., Yang, H., King, I., Lyu, M.R.: Learning latent semantic relations from clickthrough data for query suggestion. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, pp. 709–718. ACM, New York (2008)
8. Huang, C.K., Chien, L.F., Oyang, Y.J.: Relevant term suggestion in interactive web search based on contextual information in query session logs. *J. Am. Soc. Inf. Sci. Technol.* 54(7), 638–649 (2003)
9. Jones, R., Rey, B., Madani, O., Greiner, W.: Generating query substitutions. In: Proceedings of the 15th International Conference on World Wide Web, WWW 2006, pp. 387–396. ACM, New York (2006)
10. Zhang, Z., Nasraoui, O.: Mining search engine query logs for query recommendation. In: Proceedings of the 15th International Conference on World Wide Web, WWW 2006, pp. 1039–1040. ACM, New York (2006)
11. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., Vigna, S.: The query-flow graph: model and applications. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, pp. 609–618. ACM, New York (2008)
12. Bendersky, M., Croft, W.B.: Modeling higher-order term dependencies in information retrieval using query hypergraphs. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2012, pp. 941–950. ACM, New York (2012)
13. Sahami, M., Heilman, T.D.: A web-based kernel function for measuring the similarity of short text snippets. In: Proceedings of the 15th International Conference on World Wide Web, WWW 2006, pp. 377–386. ACM, New York (2006)
14. Baeza-Yates, R.: Graphs from search engine queries. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 1–8. Springer, Heidelberg (2007)
15. Kelly, D., Gyllstrom, K., Bailey, E.W.: A comparison of query and term suggestion features for interactive searching. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, pp. 371–378. ACM, New York (2009)
16. Bendersky, M., Croft, W.B.: Discovering key concepts in verbose queries. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, pp. 491–498. ACM, New York (2008)
17. Phan, X.H.: Crfchunker: Crf English phrase chunker (2006),
<http://crfchunker.sourceforge.net/>
18. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: Proceedings of the 1st International Conference on Scalable Information Systems, InfoScale 2006. ACM, New York (2006)

19. White, R.W., Bilenko, M., Cucerzan, S.: Studying the use of popular destinations to enhance web search interaction. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, Amsterdam, The Netherlands, pp. 159–166. ACM, New York (2007)
20. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2007, pp. 159–166. ACM, New York (2007)

Mining User Interest and Its Evolution for Recommendation on the Micro-blogging System

Jianjun Yu*, Yi Shen, and Jianjun Xie

Computer Network Information Center,
Chinese Academy of Sciences, 100190, China
{yujj,shenyi,xiejj}@cnic.ac.cn

Abstract. Different users have different needs, it is increasingly difficult to recommend interested topics to them. The micro-blogging system can expose user interests from individual behaviors along with his/her social connections. It also offers an opportunity to investigate how a large-scale social system recommends personal preferences according to the temporal, spatial and topical aspects of users activity. Here we focus on the problem of mining user interest and modeling its evolution on the micro-blogging system for recommendation. We learn the user preference on topics from the visited micro-boggings as user interest using text mining techniques. We then extend this concept with user's social connection on different topics. Moreover, we study the evolution of the user interest model and finally recommend the most preferred micro-boggings to a user. Experiments on a large scale of micro-blogging dataset shows that our model outperforms traditional approaches and achieves considerable performance on recommending interested posts to a user.

Keywords: recommending, user interest model, interest evolution, topic model.

1 Introduction

The micro-blogging (also named as post) allows users to exchange small elements of content such as short sentences, individual images, or video links. As a convenient communication means, especially with mobile phone, micro-blogging has been prevailing at Internet. Sina micro-blogging system would produce 25,000,000 Chinese posts each day, and Twitter get 50,000,000 posts each day. The information explosion of posts certainly brings the problem of predicting and recommending really interested micro-blogging for a user.

Recommender systems usually apply information filtering techniques to create recommendations that are indeed of users' interest. These recommender approaches seek to predict the users' "preference" on the posts they had not yet operated, using models built with the characteristics of posts, like content based approaches, or the user's interaction, like collaborative filtering approaches. Also the combined models are schemed and proved more effectively on some cases. In our opinion, the social networking functions of micro-blogging reflect users' preference to some extent, and should not be

* This research was supported by NSFC Grant No.61202408 and CNIC Innovation Grant CNIC-CX-11001.

neglected. People prefer to browse and retweet those important users' posts that concentrates on a specialized topic, i.e, the leader opinion. We think that the hybrid user interest model that combines users' social interaction on posts would demonstrate more accurate recommendations than those approaches that only rely on a single source in a micro-blogging system.

As we noticed, micro-blogging also has the characteristic that the focus changes very quickly because of the topic explosion caused by live discussion on breaking news or other themes at Internet, which also changes and influences users' interests unobtrusively and quickly. We need to consider the time factor to improve the user interest model. i.e. the evolution of user interest.

In this paper, we first define the concepts related with the user interest model, introduce the content based user interest model based on the topic model. Second, social interaction information extracted from post operations is collected to model the connection based user interest model, and then a hybrid model is schemed to calculate user preference. Finally we define two time factors modeling evolution of user interest, and recommend top N posts calculated by the approach introduced.

2 Related Work

A post in a micro-blogging system always includes two basic information: the post content and the user interaction information. Our work is closely related with two kinds of basic approaches: recommender system and social network analysis. Researchers have done much work in these fields.

Recommender systems typically produce a list of recommendations through collaborative filtering [1] or content-based filtering [2]. Modeling user interest is common practice for the construction of recommendation engines such as Amazon¹, Netflix² nowadays. Also different models are enhanced to model user interest for recommendation based on collaborative filtering [3][4], LDA [5] topic model [6], or author topic model [7]. Basically, these methods used item preference predicting user interest within the system.

Within the social network analysis area, the social information is widely used to model the user interest [8]. Also the user's social relations can be used to improve personalized social recommendation [9], boosting information sharing among users with similar interests [10], and user similarity on social media [11].

Recently, with the rising popularity of micro-blogging, much research work have been taken out leveraging both the content and connection information. [12] studied content recommendation on Twitter and explored three separate dimensions: content sources, topic interest models for users, and social voting to improve performance of recommending interesting contents. TweetSieve [13] exploited that events catch user interest gain distinguishably more attention than the average mentioning of the subject. Also, twitter based user interest model can be applied to emotion summarization [14], recommend real-time topical news [15], understand temporal dynamics of sentiment [16].

¹ <http://www.amazon.com>

² <http://www.netflix.com>

Most of work on micro-blogging introduced above were presented with user interest model considering both of textual content and social network, while the topics in micro-blogging systems are diverse and changing quickly, which brings difficulty to model users' short-term interest. Research work [17–19] presented new models to capture social network evolution [17], topic evolution [18] and community evolution [19]. These work would help us to model the user interest evolution both for the topic evolution and the social network evolution through considering time decay factors and rebuilding the user interest at each timestamp.

The work most similar to ours are [20] [21]. [20] proposed a method of making tweet recommendations based on collaborative ranking to capture personal interests. Their work considered three major elements on Twitter: tweet topic level factors, user social relation factors and explicit features such as authority of the publisher and quality of the tweet. [21] proposed a collaborative filtering approach to take user interest evolution into account. In that work, a clustering algorithm was first adopted to group the similar items. Subsequently, the preference of each cluster was calculated by the given preferences on each item in this cluster as well as the corresponding timestamps. Different from this work, we consider social networking functions to model user interest accurately since the connection information in a micro-blogging system plays the same important role with the textual content.

3 User Interest Model

As described above, our user interest model is derived from two sources: post contents and user connections. The post content means the semantics a post wants to express which is always represented with a set of keywords. The user connection is implied by the operations on the post like "following", "commenting", "retweeting". To catch a user's short-term interest without losing the long-term interests, two time factors are introduced to illustrate the principles of user interest evolution.

3.1 Content Based User Interest

Intuitively, user interest is a set of user preferences on some specialized topics, which can be abstracted from latent topic lt learned from visited posts po . That means a user u may operate different posts, a post has probability distribution on different latent topics, and thus brings the user interest on latent topics.

DEFINITION 1 (LATENT TOPIC). *A latent topic lt is what a post po is talking about. Each post can be represented with probability distribution on a set of latent topic lt : $po = \{p(lt_1), p(lt_2), \dots, p(lt_i)\}$, and each latent topic lt can be expressed with probability distribution $lt = \{p(w_1), p(w_2), \dots, p(w_i)\}$ on a set of keywords w .*

The concept of latent topic is very similar with the current prevailing topic model. Topic model is a generative model, in which a document D is presented as random mixtures over topic lt , and each topic lt is characterized by a probability distribution over keyword terms w . We adopt Latent Dirichlet Allocation (LDA) [5] to calculate the probability distribution of post po on latent topic lt .

LDA constructs a three-level hierarchical Bayesian model based on the idea of topics. We use LDA to generate each post po with multinomial dirichlet distribution over topics lt . Then repeatedly sample each topic lt with multinomial distribution over words w .

Equation (1) calculates the topic lt distribution on keywords w , and Equation (2) presents the post po distribution on topics lt .

$$p(\theta, lt, w | \alpha, \beta) = p(\theta | \alpha) \prod_{n=1}^N p(lt_n | \theta) p(w_n | lt_n, \beta) \quad (1)$$

In the three level Bayesian network of LDA, parameter α and β are applied to corpus level where α is a k vector and β is a $k \times V$ matrix (k is the dimensionality of the topic lt , V is the length of a keywords vector from a vocabulary). θ is a document level variable which presents multinomial distribution over topic lt_n and $\sum_{i=1}^k \theta_i = 1$. The lt_n and w_n are word level variables which measures the multinomial probability between a word w_n in post po with a topic lt_n .

$$p(w | \alpha, \beta) = \int p(\theta | \alpha) \left(\prod_{n=1}^N \sum_{lt_n} p(lt_n | \theta) p(w_n | lt_n, \beta) \right) d\theta \quad (2)$$

Then the user's preference on a specialized latent topic is defined as follows:

$$UIM(u_j | lt_i) = \sum_{l=1}^L P_{lt_i}^{poi} \times P_{u_j}^{poi} \quad (3)$$

where $P_{lt_i}^{poi}$ is the generative probability of latent topic lt_i distribution on the post poi calculated by Equation (1)(2), $P_{u_j}^{poi}$ is the probability that user u_j has operated on the post poi . We have

$$P_{u_j}^{poi} = \begin{cases} 1, & \text{if } poi \in \{po_{operated}\}_{u_j} \\ 0, & \text{if } poi \notin \{po_{operated}\}_{u_j} \end{cases} \quad (4)$$

After calculating user preference on each latent topic, each user has his/her preference vector on topics, we then acquire those similar neighbors for the current user.

We adopt current popular 11-distance similarity measure to estimate the user similarity of two users u_i and u_j based on the corresponding preference vector on topics. The user similarity is defined as:

$$sim_p(u_i, u_j) = 1 - \sum_{m=1}^{C_K} \| UIM(u_j | lt_m) - UIM(u_i | lt_m) \| \quad (5)$$

where C_K is the total number of the latent topics.

Before calculating the user preference on the posts, we would first introduce the connection based user interest model, and combine two kinds of user similarity functions to calculate the hybrid user preference on the posts.

3.2 Connection Based User Interest

Users' operations on posts also bring another source for recommendation. In the content based user interest model, we have got C_K topics, and then found those similar users based on these topics. In the approach of connection based user interest model, we would first define the concept of user popularity to measure the importance of each user u on each topic lt .

DEFINITION 2 (USER POPULARITY). *User popularity shows how popular a user u_i on each topic presented with a vector $UP_{u_i} = \{up_1, up_2, \dots, up_m, \dots, up_{C_K}\}$. up_m is measured with the connections between users extracted from the operations (comment, retweet, favorite, and other behaviors) on the posts.*

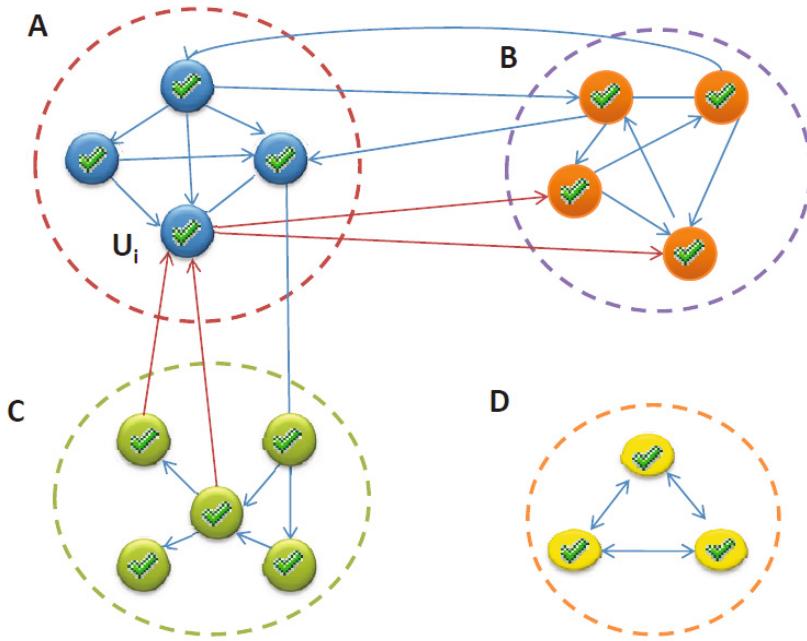


Fig. 1. The connections between users in different topics

We assume that if a post is created by a user and is operated by another user on a topic, a directed link is constructed between these two users to represent their connection. In this link, the user who made the operation would be the start point of the directed line, and the user who was operated (such as being commented, being retweeted) would be viewed as the end point of the directed line.

We give an example of the links between users in the same topic and in the different topics shown in Figure 1. For the user U_i in topic A, there are four types of connections: the start point inner topic A, the end point inner topic A, the start point to outer topic B, and the end point from outer topic C. Also two users may have dual link since they may

make operations on each other separately in topic D , we would make two connections for these dual links for simplicity.

Intuitively, we say the in degree is more important than the out degree, and the degree out of the cluster would reduce the user popularity in a topic cluster. Thus the user popularity up_m is defined as follows:

$$\widetilde{up(m)}_{u_i}^{lt_l} = D_{u_i} + \eta_1 \times \sum \frac{\overleftarrow{up(m)}_{u_k}^{lt_l}}{\overrightarrow{in(D_{u_k})}} + \eta_2 \times \sum \frac{\overrightarrow{up(m)}_{u_k}^{lt_l}}{\overleftarrow{in(D_{u_k})}} - \eta_3 \times \sum \frac{\overleftarrow{up(m)}_{u_k}^{lt_l}}{\overrightarrow{out(D_{u_k})}} - \eta_4 \times \sum \frac{\overrightarrow{up(m)}_{u_k}^{lt_l}}{\overleftarrow{out(D_{u_k})}} \quad (6)$$

where each u_k is connected with user u_i , and $\overleftarrow{in(D_{u_k})}$ is the number of connections to u_i inner cluster, $\overrightarrow{in(D_{u_k})}$ is the number of connections from u_i inner cluster, $\overleftarrow{out(D_{u_k})}$ is the number of connections to u_i out of cluster, $\overrightarrow{out(D_{u_k})}$ is the number of connections from u_i out of cluster.

To make $up(m)_{u_i}^{C_l}$ be a value between 0 and 1, we would normalize the value $\widetilde{up(m)}_{u_i}^{C_l}$ divided with $(D_{u_i} + \eta_1 \times \overleftarrow{in(D_{u_k})} + \eta_2 \times \overrightarrow{in(D_{u_k})} - \eta_3 \times \overleftarrow{out(D_{u_k})} - \eta_4 \times \overrightarrow{out(D_{u_k})})$.

Then each user has his/her personal popularity vector on topics. We adopt the cosine similarity measure to estimate the user similarity of users u_i and u_j based on the corresponding popularity vectors. The cosine similarity measure is defined as:

$$sim_u(u_i, u_j) = \frac{\sum up_i \times up_j}{\sqrt{\sum (up_i^2)} \times \sqrt{\sum (up_j^2)}} \quad (7)$$

3.3 Combined User Interest

In this paper, user interest would be modeled with the content of post and connection of users. We combine Equation (5) and (7) to provide the hybrid similarity of users as presented with Equation (8).

$$sim(u_i, u_j) = \mu sim_p(u_i, u_j) + (1 - \mu) sim_u(u_i, u_j) \quad (8)$$

We then identify top N_C nearest neighbors for the user according to Equation (8). With the top N_C neighbors, the user preference on a post po is defined as follows.

$$UIM_u^{po} = V(UIM_{u_i}^{po_m}) + \frac{\sum_{j=1}^{N_C} V(UIM_{u_j}^{po_m}) \times sim(u_i, u_j)}{sim(u_i, u_j)} \quad (9)$$

where $V(UIM_{u_i}^{po_m})$ is measured with Equation (10).

$$V(UIM_{u_i}^{po_m}) = UIM(u_i | lt_l) \times (1 - P_{u_i}^{po_m}) \times P_{lt_k}^{po_m} \quad (10)$$

$P_{u_i}^{po_m}$ is the probability that user u_i has operated on the post po_m as described by Equation (4). $P_{lt_k}^{po_m}$ is the post probability distribution on latent topic, which can be calculated by Equation (1)(2).

We then generate the recommendations selecting top N_p posts with the highest values of UIM_u^{po} to a user.

3.4 User Interest Evolution

Micro-blogging is a place to share those latest news and information, which also brings the issue of changing focuses very quickly for a user. For example, a user would concentrate on "social network analysis" at this moment, and "cooking" at next moment. These two topics are totally unrelated, whereas they illustrate real interest change for a user at different periods.

We made an investigation on two users concentrating their interest change from 01/01/2012 to 31/03/2012. User u_1 is a software engineer, he focuses on the topics related with his work at timestamp 01/01-10/01. He then will change his interest since he plans to travel for his Spring Festival at the next timestamp. Finally he will change his interest again when he is back to work. User u_2 is a computer scientist, he is interested in the topic "gene search" since he would write a conference paper related to this topic with his novel data mining algorithm. He would finally present the same interest as the one at the first period since he has finished this conference paper.

We define two time factors to model user interest evolution: decay factor π_{df} and near time factor π_{ntf} . Decay factor aims to reduce the impact of the post which is published or discussed long time ago. Near time factor aims to increase the impact of the post which is published or discussed at current time.

Consequently, to calculate $V(UIM_{u_i}^{po_m})$, we should consider the decay factor π_{df} and near time factor π_{ntf} which can be measured as follows:

$$\begin{cases} \pi_{df} = \frac{\Delta t}{t_i - t_0} \\ \pi_{ntf} = 1 - w_t \\ V(UIM_{u_i}^{po_m})_{new} = V(UIM_{u_i}^{po_m}) \times \pi_{df} \times \pi_{ntf} \end{cases} \quad (11)$$

where Δt is a timestamp to cut posts to different periods, t_0 is labeled as the initial time and set as 01/10/2011 in this paper, t_i is the operation time on the post. w_t is measured as follows:

$$w_t = \begin{cases} \log_{N_{max}} \text{cell}\left(\frac{t_{now}-t_i}{\Delta t}\right) & \text{if } \text{cell}\left(\frac{t_{now}-t_i}{\Delta t}\right) \leq N_{max} \\ 1 & \text{if } \text{cell}\left(\frac{t_{now}-t_i}{\Delta t}\right) > N_{max} \end{cases} \quad (12)$$

N_{max} is the maximum value that aims to omit those overdue posts long time ago.

We then apply the above equations to update the user preference on the post after each timestamp Δt .

3.5 Recommending

Recommending is a process of selecting top N_p posts with highest preference values calculated by our user interest model. Considering the calculation of user preference is a time consuming process, we prefer to execute the algorithm at each timestamp Δt .

4 Experimental Evaluation

We crawled 1,201,799 original micro-bloggings from Sina Weibo micro-blogging system with its API from 2011/10/01 to 2012/3/31. There are 726,552 new posts, 475,247 retweets, 506,239 comments, 456,720 @, 19,982 tags, and 10,300 favorites from 3,312 users. In the collected dataset, we choose our training dataset from 2011/10/01 to 2011/11/30, and the remains as the test dataset. In this training dataset, there are 304,160 micro-bloggings and 914,873 words identified by Chinese terms. Considering the particularity of the Chinese micro-blogging system, we generated these Chinese terms from several basic corpus, including Sogou Pinyin input dict³, nlpirc micro-blogging corpus⁴. Moreover, to avoid the possible bias of training user preference, we randomly choose 50 users from our dataset, and "@" them with those individual recommending micro-bloggings. The volunteers showed their best effort helping to retweet those preferred micro-bloggings.

We adopt the evaluation criteria precision to evaluate the prediction accuracy, which is widely used in the information retrieval area and defined as follows:

$$\text{precision} = \frac{\text{hits}}{N_p} \quad (13)$$

where *hits* presents the number of recommending posts that are labeled as interested ones, and N_p is the total recommendations produced by our approach.

4.1 Parameters Tuning for User Interest Model

Our model would involves several parameters, including the number of topics C_K , the weight of connection η , the weight of similarity μ , the timestamp of interest change Δt , the maximum value to omit overdue posts N_{max} , the number of neighbors N_C , and the number of recommending results N_p . Because these parameters interact with each other and effect the final result of the recommending precision, we would just fix the other parameters when doing the experiment on one parameter for simplicity. Finally, we got the tuning values for each parameter to acquire the best recommending precision as shown in Table 1.

4.2 The Precision of Our Approach

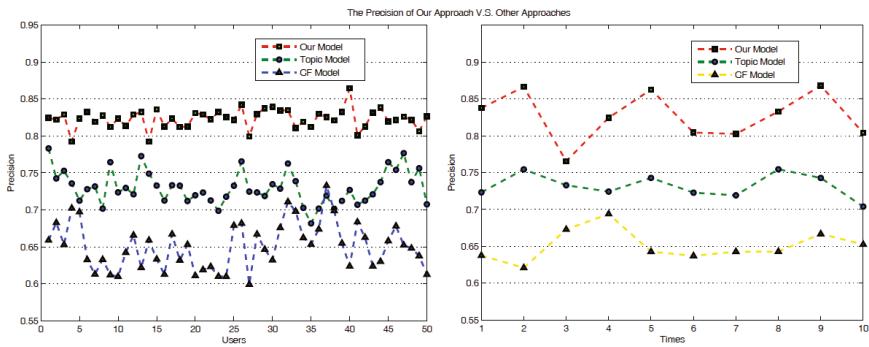
To check the precision of our approach, We repeated the experiment 10 times for each 10 days to get the average result. Also the comparisons of our model with the topic

³ <http://pinyin.sogou.com/dict>

⁴ <http://www.nlpirc.org>

Table 1. Parameters tuning results for user interest model

Parameter	Description	Value
C_K	the number of topics	200
η_1	the weight of in degree inner topic	0.8
η_2	the weight of out degree inner topic	0.4
η_3	the weight of in degree outer topic	0.3
η_4	the weight of out degree outer topic	0.2
μ	the weight of similarity	0.6
N_C	the number of neighbors	20
Δt	the timestamp of interest change	10
N_{max}	the maximum value to omit overdue posts	180
N_p	the number of recommending results	15

**Fig. 2.** The precision of our approach V.S. other approaches

model and the collaborative filtering approach were provided. The topic model approach aims to model the user interest with the content information, and the collaborative filtering approach aims to model the user interest with the connection information.

As shown in Figure 2, we can see that our approach outperforms than the topic model and the collaborative filtering approaches since our approach considers the factors of the content and the connection information, and also brings the concept of evolution of interest. The topic model approach would achieve better precision than the collaborative filtering approach because there doesn't exist the rank score for micro-boggings, and the "browser" operations are missing since we cannot get the visiting logs.

In the left side of Figure 2, we got the average result of 10 experiments for each user. We found that our model is better than the other approaches, and the recommending precision is more smooth than the others' because our model considered the factors of topic classification, user relation and topic evolution though there exists influence from subjective evaluation. In the right side of Figure 2, we got the average precision of 50 users for each recommending time. We found that though our result is better than the other approaches', each period exhibits apparently different precision. One reason is that the topic we classified may not reflect the real interested topic, such as different topic scope, topic hotness, and topic life cycle, which would be discussed in the next

section. In our opinion, our model is suitable for general recommendation, still we need to consider more factors to keep the stability of recommending precision.

5 Discussion

As we described above, each user exhibits different precision of recommending for the reason of topic scope, topic hotness, topic life cycle. We then gave an investigation on these factors that reflect the influence of recommending precision. We first selected four specified topics to check the precision results on "computer science", "breaking news", "travel", and "food", which had the same experimental steps as the above section.

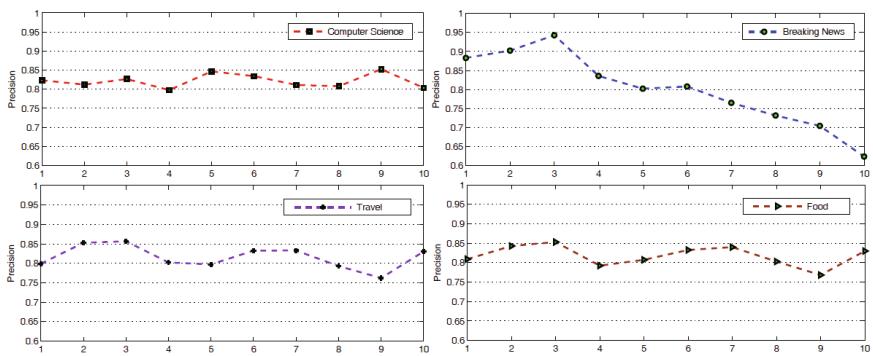


Fig. 3. The precision of different topics

As shown in Figure 3, recommending topics on "computer science" presents almost the same result of the general recommending in the right side of Figure 2, this is because the topic "computer science" is also a big topic that composes of many sub topics. Recommending topics on "breaking news" presents acute fluctuation of recommending precision, this is because the topics on breaking news always have short concentration time but present strong hotness. This phenomenon would bring user interest on the topics for a short time but with a strong concentration. The topics "travel" and "food" present almost the same influence of recommending precision, this is because when a user travels to some palaces, he/she would also interest local delicious foods meanwhile. In the real recommending system, we should take the following factors to improve the commanding precision: 1) life cycle and hotness of specialized topics, such as breaking news, holiday related topics; 2) the relations of topics, such as "travel" and "food".

Considering that the topic "computer science" is too big to model real user interest, we selected two sub topics "cloud computing", and "recommender system" to check the precision. We repeated the experiments introduced in the above section expect that we just selected 10 users because few users interested on the topic "recommender system".

The upper two figures in Figure 4 presented two latent information for recommendation. The first one is that the precision of "recommender system" is higher than the one of "cloud computing", and the connection based user interest model provides more

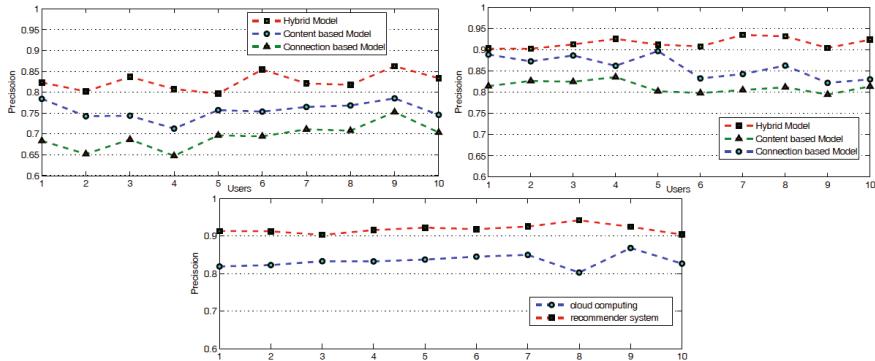


Fig. 4. The precision of different sub topics on "computer science"

contribution on the precision. We found that the users interested in topic "recommender system" are relatively professional and fixed, and their connections are tight, which would exhibit higher precision. Whereas "cloud computing" is more prevailing, many users involved in this topic, and the connections are loose, which would present lower precision as a whole, and the precision of content based user interest model is higher than the one of connection based user interest model. The second one is that the precision on "recommender system" is more smooth and higher than the one on "cloud computing" because the users interested on the first topic are always professional users, form a small team, and present the tight connection to discuss those most interested micro-bloggings. In the real recommending system, we should take the following factors to improve the commanding precision: 1) the number of interest users and their reputation; 2) the popularity of topics, for example, the topic "cloud computing" may be more popular than the topic "recommender system" in the micro-blogging system.

6 Conclusion

The micro-blogging system shows its rapid dissemination speed of updated news and topics, so how to recommend preferred items to a user from large scale of micro-bloggings according to his/her interest is a research focus. In this paper, we scheme a hybrid user interest model that combines the content and the social interaction information to calculate the user preference on the micro-bloggings, and define two time factors modeling user interest evolution.

Still there are several issues should be solved. The first one is that a user may change his/her interest smoothly, which can be viewed as "topic evolution", we should make a further investigation on this topic evolution behavior to model user's interest more accurately. The second problem is that retweets and comments present strong attitude on the post. The attitude may be his or her judgement or evaluation, affective state, or the intended emotional communication to represent his/her strong interest or uninterest. We need to adopt opinion mining approach to identify these subjective information.

References

1. Koren, Y.: Collaborative filtering with temporal dynamics. *Communications of the ACM* 53(4), 89–97 (2010)
2. Mooney, R.J., Roy, L.: Content-based book recommendation using learning for text categorization. In: Proc. DL 2000, pp. 195–240. ACM Press (2000)
3. Hannon, J., Bennett, M., Smyth, B.: Recommending twitter users to follow using content and collaborative filtering approaches. In: Proc. RecSys 2010, pp. 199–206. ACM Press (2010)
4. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proc. KDD 2008, pp. 426–434. ACM Press (2008)
5. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. *The Journal of Machine Learning Research* 3(3), 993–1022 (2003)
6. Chen, W.Y., Chu, J.C., Luan, J., et al.: Collaborative filtering for orkut communities: discovery of user latent behavior. In: Proc. WWW 2009, pp. 681–690. ACM Press (2009)
7. Xu, Z., Lu, R., Xiang, L., et al.: Discovering user interest on twitter with a modified author-topic model. In: Proc. WI-IAT 2011, pp. 422–429. IEEE Computer Society (2011)
8. Wen, Z., Lin, C.Y.: Improving user interest inference from social neighbors. In: Proc. CIKM 2011, pp. 1001–1006. ACM Press (2011)
9. Carmel, D., Zwerdling, N., Guy, I., et al.: Personalized social search based on the user’s social network. In: Proc. CIKM 2009, pp. 1227–1236. ACM Press (2009)
10. Shepitzen, A., Gemmell, J., Mobasher, B., et al.: Personalized recommendation in social tagging systems using hierarchical clustering. In: Proc. RecSys 2008, pp. 259–266. ACM Press (2008)
11. Anderson, A., Huttenlocher, D., Kleinberg, J., et al.: Effects of user similarity in social media. In: Proc. WSDM 2012, pp. 703–712. ACM Press (2012)
12. Nairn, J.C.R., Nelson, L., et al.: Same places, same things, same people?: mining user similarity on social media. In: Proc. CHI 2010, pp. 1185–1194. ACM Press (2010)
13. Grinev, M., Grineva, M., Boldakov, A., et al.: Sifting micro-blogging stream for events of user interest. In: Proc. SIGIR 2009, p. 837. ACM Press (2009)
14. Li, C.T., Lai, H.C., Ho, C.T., et al.: Pusic: musicalize microblog messages for summarization and exploration. In: Proc. WWW 2010, pp. 1141–1142. ACM Press (2010)
15. Phelan, O., McCarthy, K., Smyth, B.: Using twitter to recommend real-time topical news. In: Proc. RecSys 2009, pp. 385–388. ACM Press (2009)
16. Diakopoulos, N.A., Shamma, D.A.: Characterizing debate performance via aggregated twitter sentiment. In: Proc. CHI 2010, pp. 1195–1198. ACM Press (2010)
17. Zheleva, E., Sharara, H., Getoor, L.: Co-evolution of social and affiliation networks. In: Proc. KDD 2009, pp. 1007–1016. ACM Press (2009)
18. Lin, C.X., Zhao, B., Mei, Q., et al.: Pet: a statistical model for popular events tracking in social communities. In: Proc. KDD 2010, pp. 929–938. ACM Press (2010)
19. Tang, L., Liu, H., Zhang, J., et al.: Community evolution in dynamic multi-mode networks. In: Proc. KDD 2008, pp. 677–685. ACM Press (2008)
20. Chen, K., Chen, T., Zheng, G., et al.: Collaborative personalized tweet recommendation. In: Proc. SIGIR 2012. SIGIR 2012 Proceedings (2012)
21. Cheng, T.H., Chen, H.C., Lin, W.B., et al.: Collaborative filtering with user interest evolution. In: Proc. PACIS 2011. PACIS 2011 Proceedings (2011)

Collaborative Filtering Using Multidimensional Psychometrics Model

Haijun Zhang¹, Xiaoming Zhang¹, Zhoujun Li¹, and Chunyang Liu²

¹ School of Computer Science and Engineering, Beihang University, Beijing, China, 100191

² National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029

haijun_cumtb@yahoo.com.cn, {yolixs,lizj}@buaa.edu.cn,
lcy@isc.org.cn

Abstract. In this paper, the psychometrics model, i.e. the rating scale model, is extended from one dimension to multiple dimension. Then, based on this, a novel collaborative filtering algorithm is proposed. In this algorithm, user's interest and item's quality are represented by vectors. User's rating for an item is a weighted summation of the user's latent ratings for the item in all dimensions, in which the weights are user-specific. Moreover, user's latent rating in each dimension is assumed to follow a multinomial distribution that is determined by the user's interest value, the item's quality value in this dimension, and the thresholds between two consecutive ratings. The parameters are estimated by minimizing the loss function using the stochastic gradient descent method. Experimental results on the benchmark data sets show the superiority of our algorithm.

Keywords: Psychometrics, Rating Scale Model, Collaborative Filtering, Latent Interests, Stochastic Gradient Descent.

1 Introduction

Memory-based collaborative filtering (CF) is a popular method to predict user's preference. The key step of memory-based CF is the calculation of similarity between users or items. Data sparsity makes the computed similarity incredible and hence results in low prediction accuracy. Biyun Hu et al. [1,2] introduced the psychometrics model into memory-based CF to alleviate this problem and achieved better results. However, they assume that each user has only one interest, which does not conform to the actuality, because user usually has interests in many aspects or fields. On the other hand, they uses Off-the-shelf software package — Winsteps [3] to learn the parameters, which limits its flexibility and scalability. To solve this problem, in this paper, a collaborative filtering algorithm based on multidimensional psychometrics model (CFMPM) is proposed. The parameters are estimated by stochastic gradient descent method.

The rest of this paper is organized as follows. Section 2 describes the psychometrics model. In section 3, CFMPM is introduced. Two experiments are designed and analyzed in section 4, followed by conclusions in section 5.

2 Psychometrics Model

In psychometrics, latent trait models also known as item response theory, are a series of mathematical models for measuring examinee's latent trait, such as ability, interest or attitude by his or her response to one question/item or more. Traditional latent trait model was proposed by Rasch [4] in 1960, which can be expressed as Eq. (1).

$$\log \left(\frac{p(r_{ui}=1)}{p(r_{ui}=0)} \right) = B_u - D_i \quad (1)$$

where, B_u is the ability value of examinee u , and D_i is the difficulty value of question i , and $p(r_{ui}=1)$ is the probability that examinee u will succeed on question i , and $p(r_{ui}=0)$ is the probability that examinee u will get a failure on question i . Traditional Rasch model can only deal with binary values (success or failure, usually coded by 1 or 0). Andrich [5] extended Rasch model to handle multivalued scores, which is called rating scale model. It can be expressed as Eq. (2) and (3).

$$\log \frac{P_{uik}}{P_{ui(k-1)}} = B_u - D_i - F_k \quad k = 2, 3, \dots, K \quad (2)$$

$$F_1 = 0 \quad (3)$$

where P_{uik} is the probability of examinee u scores k points on question i , F_k is the ordered threshold denoting the difficulty of getting k points relative to getting $k-1$ points, and each question has K -level scores, i.e. 1, 2, ..., K . B_u , D_i , and $F_2 \sim F_K$ are the parameters need to be estimated.

3 Our Framework

In this section, we will extend psychometrics model from one dimension to multi-dimension. Based on that, a novel CF model and a parameter learning method are proposed. And then, the workflow of the algorithm is described.

3.1 Collaborative Filtering Based on Multidimensional Psychometrics Model

In rating scale model expressed by Eq.(2), user's ability B_u and item's difficulty D_i are scalar quantities, which denote the user's interest and the item's quality in CF [1,2] respectively. However, it is more reasonable to represent user's interest and item's quality by vectors. Based on this idea, the psychometrics model is extended to multi-dimension, which is expressed as Eq. (4) and (5).

$$\log \frac{P_{uikl}}{P_{ui(k-1)l}} = B_{ul} - D_{il} - F_k \quad k = 2, 3, \dots, K; l = 1, 2, \dots, L \quad (4)$$

$$F_1 = 0 \quad (5)$$

where, P_{uikl} is the probability of user u gives rating score k to item i in dimension l , and B_{ul} is the interest value of user u in dimension l , and then $B_u = (B_{u1}, B_{u2}, \dots, B_{uL})^T$ is the interest vector of user u . High value of B_{ul} means user u has strong interest in dimension l , which implies that if a film can satisfy user's interest in this aspect to some degree, he or she will give a high latent rating value for this film in this dimension. While D_{il} is the quality value of item i in dimension l , and thus $D_i = (D_{i1}, D_{i2}, \dots, D_{iL})^T$ is the quality vector of item i . Bigger D_{il} means item i has less content related to dimension l , and it has a smaller probability to get a high rating in dimension l , and F_k is the ordered threshold measuring the difficulty for all users to give k points relative to give $k-1$ points to an item. Each item has K -level ratings. In MovieLens¹ data set, K is set to 5. L is the number of dimensions which denotes the number of latent interests/demands each user has, and it is a meta-parameter, which should be set in advance. If L is set to 1, this model will degenerate into the model represented by Eq. (2). B_{ul} , D_{il} , and F_k are parameters need to be estimated. For easy description, a notation ψ_k is introduced, and then Eq. (6)-(8) can be deduced from Eq. (4)-(5):

$$\psi_k = -\sum_{j=1}^k F_j \quad (6)$$

$$\psi_1 = 0 \quad (7)$$

$$P_{uikl} = \frac{e^{\psi_k + (k-1)(B_{ul} - D_{il})}}{\sum_{j=1}^K e^{\psi_j + (j-1)(B_{ul} - D_{il})}} \quad k = 1, 2, \dots, K; \quad l = 1, 2, \dots, L \quad (8)$$

In Eq. (8), it can be seen that the rating of user u for item i in dimension l is of multinomial distribution with K categories which is determined by B_{ul} , D_{il} , and ψ_k . Then, the latent rating of user u for item i in dimension l (denoted by \hat{r}_{uil}) can be estimated by Eq.(9). After that, the rating value of user u for item i can be computed by Eq.(10).

$$\hat{r}_{uil} = \sum_{k=1}^K k \cdot P_{uikl} \quad (9)$$

$$\hat{r}_{ui} = \frac{\sum_{l=1}^L w_{ul} \cdot \hat{r}_{uil}}{\sum_{l=1}^L w_{ul}} \quad (10)$$

where $w_{ul} \geq 0$, which denotes the weight of user u in dimension l , and it represents the importance of the l -th interest/demand for user u , and then the weight vector of user u can be denoted by $w_u = \{w_{u1}, w_{u2}, \dots, w_{uL}\}^T$, and it will be estimated using training set.

¹ <http://www.grouplens.org/>

3.2 Parameters Learning Method

In order to learn the parameters, the loss function is designed as Eq. (11).

$$f = \sum_{(u,i) \in Tr} (r_{ui} - \hat{r}_{ui})^2 + \lambda_1 \sum_{u,l} w_{ul}^2 + \lambda_2 \sum_{u,l} B_{ul}^2 + \lambda_3 \sum_{i,l} D_{il}^2 + \lambda_4 \sum_j \psi_j^2 \quad (11)$$

where Tr denotes the training set. The first term on the right side of the formula is the sum of square error, and the other four terms are the regularization terms that are applied to parameters to avoid over-fitting. The regularization constants λ_1 , λ_2 , λ_3 , and λ_4 can be determined by cross validation.

We use stochastic gradient descent method [6] to learn the parameters. Each rating r_{ui} in training set Tr is used to update the parameters according Eq. (14)-(17). Parameter ψ_l should not be updated and it is fixed as 0 according to Eq. (7). Learning ratio parameters γ_1 , γ_2 , γ_3 , and γ_4 are determined by cross validation. For easy representation, notations e_{ui} and ψ_{jul} are introduced.

$$e_{ui} = r_{ui} - \hat{r}_{ui} \quad (12)$$

$$\psi_{jul} = e^{\psi_j + (j-1).(B_{ul} - D_{ul})} \quad \text{for } j=1,2,\dots,K; l=1,2,\dots,L \quad (13)$$

$$w_{ul} \leftarrow \max \left(0, w_{ul} + \gamma_1 \left(e_{ui} \cdot \left(\frac{\hat{r}_{uil} \cdot \sum_{l=1}^L w_{ul} - \sum_{l=1}^L (w_{ul} \cdot \hat{r}_{uil})}{\left(\sum_{l=1}^L w_{ul} \right)^2} \right) - \lambda_1 w_{ul} \right) \right) \quad (14)$$

$$B_{ul} \leftarrow B_{ul} + \gamma_2 \left(\frac{e_{ui}}{\sum_{l=1}^L w_{ul}} \cdot w_{ul} \cdot \left(\sum_{k=1}^K k \cdot \frac{(k-1) \cdot \psi_{kul} \cdot \left(\sum_{m=1}^K \psi_{mul} \right) - \psi_{kul} \cdot \left(\sum_{m=1}^K (m-1) \cdot \psi_{mul} \right)}{\left(\sum_{m=1}^K \psi_{mul} \right)^2} \right) - \lambda_2 B_{ul} \right) \quad (15)$$

$$D_{ul} \leftarrow D_{ul} + \gamma_3 \left(\frac{e_{ui}}{\sum_{l=1}^L w_{ul}} \cdot w_{ul} \cdot \left(\sum_{k=1}^K k \cdot \frac{(1-k) \cdot \psi_{kul} \cdot \left(\sum_{m=1}^K \psi_{mul} \right) - \psi_{kul} \cdot \left(\sum_{m=1}^K (1-m) \cdot \psi_{mul} \right)}{\left(\sum_{m=1}^K \psi_{mul} \right)^2} \right) - \lambda_3 D_{ul} \right) \quad (16)$$

$$\psi_j \leftarrow \psi_j + \gamma_4 \left(\frac{e_{ui}}{\sum_{l=1}^L w_{ul}} \cdot \sum_{l=1}^L w_{ul} \cdot \frac{j \cdot \psi_{jul} \cdot \left(\sum_{m=1}^K \psi_{mul} \right) - \left(\sum_{m=1}^K m \cdot \psi_{mul} \right) \cdot \psi_{jul}}{\left(\sum_{m=1}^K \psi_{mul} \right)^2} - \lambda_4 \psi_j \right) \quad \text{for } j=2,\dots,K \quad (17)$$

3.3 Workflow of the Algorithm

The pseudo codes of algorithm CFMPM are shown in figure 1. In the two experiments in section 4, the initial values of B_{ul} and D_{ul} are sampled from uniform distribution on

$[0, 0.5]$, and the initial value of w_{ul} is set to 1, and ψ_k ($k=1, 2, \dots, K$) is set to 0. The learning ratio constants $\gamma_1, \gamma_2, \gamma_3$, and γ_4 are set to values between 0.015 and 0.5. The regularization constants $\lambda_1, \lambda_2, \lambda_3$, and λ_4 are set to 0.001. After the parameters are obtained by the algorithm, the ratings in test set can be predicted by Eq. (10).

-
1. Input: training set Tr
 2. Output: B_u, D_i, w_u , and ψ_k ($k=2, 3, \dots, K$)
 3. Initialize parameters $B_u, D_i, w_u, \psi_k, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2, \lambda_3$, and λ_4 .
 4. For each iteration
 5. Randomly permute the data in Tr .
 6. For each r_{ui} in Tr
 7. Step 1. Compute e_{ui} by Eq.(12);
 8. Step 2. Compute ψ_{jul} ($j=1, 2, \dots, K; l=1, 2, \dots, L$) by Eq.(13);
 9. Step 3. Update the parameters as shown in Eq. (14)-(17).
 10. End for
 11. Decrease $\gamma_1, \gamma_2, \gamma_3, \gamma_4$; they are multiplied by 0.9.
 12. End for
-

Fig. 1. Algorithm CFMPM

4 Experiments

The MovieLens 100k data set is used to evaluate the algorithm CFMPM.

4.1 Comparison with One-Dimensional Psychometrics Model Based CF

To compare CFMPM with the CF algorithms based on one-dimensional psychometrics model which are proposed in [1, 2] (denoted by Alg1 and Alg2), 5-fold cross validation was done. In this experiment, the interest dimensionality L is set to 25, and when the difference of MAE on the training set between two consecutive iterations is less than 0.00005, the iteration is terminated. Comparison results are shown in table 1.

Table 1. Comparison on MAE with the Algorithms Reported in [1,2]

Algorithm	CFMPM	Alg1	Alg2
MAE	0.7249	>0.80	0.725

Although the MAE of CFMPM is not significant lower than that of Alg2, Alg2 fuses memory-based CF and computes all similarities between every two users and thus its scalability is decreased. So, it means that extending psychometrics model from one dimension to multi-dimension is helpful.

4.2 Comparison with Other State-of-the-Art Algorithms

To consistently compare algorithm CFMPM with the state-of-the-art CF algorithms reported in [7,8], as they did, we also extract a subset which contains 1000 items and

500 users from MovieLens data set, where each of the users has more than 40 ratings (data density is 10.35%). The first 100, 200 and 300 users in the data set are selected to build three different training sets respectively, and they are denoted as ML_100, ML_200 and ML_300. But for different training sets, the test set is fixed, i.e., it is set with the last 200 users. In order to evaluate the performance of the algorithm in different data sparsity level, we also randomly selected 5, 10 and 20 ratings of the test users into the observed set, and they are named Given5, Given10, and Given20 respectively, while the other ratings of the test users constitute the held out set.

Dimensionality L in CFMPM is set to 20, and the iteration times is set to 10. As it is done in [7], for every training set on every sparsity level, the observed data set of test users are randomly selected 10 times. The reported results in table 2 are the average results over 10 times. From table 2, it can be seen that CFMPM almost outperforms all the other algorithms on every training set at every data sparsity level.

Table 2. Comparison on MAE with the Algorithms Reported in [7,8]

Training Set	Algorithms	Given5	Given10	Given20
ML_100	CFMPM	0.785	0.772	0.756
	CFONMTF	0.838	0.801	0.804
	CBT	0.840	0.802	0.786
	CBS	0.874	0.845	0.839
	WLR	0.915	0.875	0.890
	SF2	0.847	0.774	0.792
	PCC	0.874	0.836	0.818
ML_200	CFMPM	0.781	0.768	0.753
	CFONMTF	0.827	0.791	0.787
	CBT	0.839	0.800	0.784
	CBS	0.871	0.833	0.828
	WLR	0.941	0.903	0.883
	SF2	0.827	0.773	0.783
	PCC	0.859	0.829	0.813
ML_300	CFMPM	0.780	0.764	0.751
	CFONMTF	0.801	0.780	0.782
	CBT	0.840	0.801	0.785
	CBS	0.870	0.834	0.819
	WLR	1.018	0.962	0.938
	SF2	0.804	0.761	0.769
	PCC	0.849	0.841	0.820

5 Conclusions

Rating an item for a user is a psychological process. Thus, studying psychology models for the purpose of CF is meaningful. The algorithm CFMPM demonstrates this point. Moreover, CFMPM can be fused in other method to improve the performance

further. Meanwhile, when we increase the dimensionality L , usually we can get a decreased MAE, but if the L is increased further and further, the MAE will increase, and the parameters must be tuned more carefully to avoid over-fitting, so, how to choose the dimensionality automatically is a problem we will study in future.

Acknowledgement. This work was supported by the National Natural Science Foundation of China [No. 60973105, 90718017, 61170189, and 61202239], the Research Fund for the Doctoral Program of Higher Education [No. 20111102130003] and the Fund of the State Key Laboratory of Software Development Environment [No. KLSDE-2011ZX-03].

References

1. Hu, B., Li, Z., Wang, J.: User's latent interest-based collaborative filtering. In: Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., van Rijsbergen, K. (eds.) ECIR 2010. LNCS, vol. 5993, pp. 619–622. Springer, Heidelberg (2010)
2. Hu, B., Li, Z., Chao, W., et al.: User preference representation based on psychometric models. In: Proc. of 22nd Australasian Database Conference (2011)
3. Linacre, J.M.: WINSTEPS: Rasch measurement computer program, Chicago (2007), <http://Winsteps.com>
4. Rasch, G.: Probabilistic models for some intelligence and attainment tests. Institute of Educational Research, Copenhagen (1960)
5. Andrich, D.: A rating formulation for ordered response categories. *Psychometrika* 43, 561–573 (1978)
6. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proc. of KDD (2008)
7. Li, B., Yang, Q., Xue, X.: Can movies and books collaborate?—Cross-domain collaborative filtering for sparsity reduction. In: Proc. of IJCAI, pp. 2052–2057 (2009)
8. Chen, G., Wang, F., Zhang, C.: Collaborative filtering using orthogonal nonnegative matrix tri-factorization. *Information Processing & Management* 45(3), 368–379 (2009)

Incorporating Social Actions into Recommender Systems

Di Ma, Dandan Song*, and Lejian Liao

Beijing Engineering Research Center of High Volume language Information Processing & Cloud Computing Application,
Beijing Lab of Intelligent Information Technology,
School of Computer Science, Beijing Institute of Technology, Beijing, China
madi171@gmail.com, {sdd, liaolj}@bit.edu.cn

Abstract. With the rapidly growing amount of information available on the internet, recommender systems become popular tools to promote relevant online information to a given user. Although collaborative filtering is the most popular approach to build recommender systems and has been widely deployed in many applications, it still pay little attention to social actions, which are widely common in social networks and we believe could make a significant improvement in recommender systems. In this paper, we incorporate users' social actions into a model-based approach for recommendation using probabilistic matrix factorization. Compared with previous work, users' social actions are taken as a new relation to optimize previous trust-based recommender systems. To achieve this, we propose a social recommendation graphical model employing users' relations based on their social actions. We make use of users' commenting action in our approach and conduct experiments on a real life dataset, extracted from the Douban movie ratings and comments system. Our experiments demonstrate that incorporating users' social action information leads to a significant improvement in recommender systems.

Keywords: Recommendation, Social Network, Social Actions, Probabilistic Matrix Factorization.

1 Introduction

In the past decades of years, recommender systems have been widely studied in the academia and deployed in the industry, such as Amazon Online Shop and Yahoo! Music etc. However, there are still some challenges in traditional recommender systems. The well known one is that most modern commercial recommender systems are facing serious sparsity and cold-start problem as the average density of the available ratings is often less than 1%.

In our real life online shopping, people always tend to ask their friends for recommendations of nice mobile phone or fantastic movies, or view product's

* Corresponding author.

or movie's comments to make decisions, we are actually doing social recommendations everyday in our normal life. Therefore, these social actions are precious information for obtaining people's interests, which could provide more personalized recommendation and improve recommender systems.

In this paper, we aim at solving the sparsity and cold-start problems with social actions information and modeling a better performance recommender system. Based on the intuition that user's trust relations can be employed to enhance traditional recommender systems, our approach assumes a social network among users and makes recommendations for a user based on the ratings of the users that have direct or indirect social relations with the given user. Therefore, we believe that users' social action information can provide more valuable clue to improve collaborative filtering performance. Our model works under an assumption that actions between users on the social network are conducted if they share very similar values and interests. The more frequently they interact, the higher similarity their interests share.

We propose a social recommendation graphical model employing users' relations based on their social actions. The social network information including friends' relations and commenting interactions are employed in designing a new social recommender system with probabilistic matrix factorization. Our experiments were performed on the dataset that we crawled from Douban¹.

The rest of this paper is organized as follows: Some related work is discussed in section 2. A simple description of social recommender framework and an introduction to our proposed model is given in section 3. Experimental result is shown in section 4. Finally, we conclude the paper and present some directions for future work.

2 Related Work

In this section, we review several major approaches for recommender systems, especially for collaborative filtering. Generally two type of recommender systems have been investigated: Memory-based and Model-based recommenders.

Memory-based approaches are the most popular prediction methods and are widely adopted in commercial collaborative filtering systems. Memory-based algorithms explore the user-item rating matrix and make recommendations based on the ratings of item i by a set of users whose rating profiles are most similar to that user u . The most analyzed examples of memory-based collaborative filtering include user-based approaches and item-based approaches.

On the contrary, model-based approaches learn the parameters of a model and store only those parameters. Hence they do not need to explore the rating matrix and only store the model parameters. Model-based approaches are very fast after the parameters of the model are learnt. The bottleneck for model-based approaches is the training phase, while in memory-based approaches there is no

¹ www.douban.com, is a social networking service in which users can perform various social actions, such as rating movies, making movie reviews and creating their own social network.

training, but the prediction (test) phase is time-consuming. Major model-based approaches include the clustering model [1], aspect models [2], the Bayesian model [3] and the most famous latent factor model [4,5]. Recently, some matrix factorization methods [6,3,7,8] have been proposed for collaborative filtering. These methods all focus on fitting the user-item rating matrix using low-rank approximations, and use it to make further predictions. Recently, due to the efficiency in dealing with large datasets, several low-dimensional matrix factorization methods have been proposed for Collaborative filtering [8]. The premise behind the low-rank factor model is that there is only a user's preference vector determined by how each factor applies to that user.

All these methods mentioned above do not agree with a real life recommendation as they care little about social network information. Actually, many researchers have recently started to analyze trust-based recommender systems based on this intuition. In [9,10], a social trust ensembled collaborative filtering method for recommender systems is proposed.

3 Social Recommendation Model

3.1 Recommendations with Social Actions

In this section, we present our approach to incorporate social action relation into a matrix factorization model for recommendations in social network.

The author of [10] proposed a matrix factorization approach for social network based recommendation, called STE. Their method is a linear combination of basic matrix factorization approach [7] and a social network based approach. The predicted rating of user u on item i is as follows:

$$R_{u,i} = g(\alpha U_u^T V_i + (1 - \alpha) \sum_{v \in N_u} T_{u,v} U_v^T V_i), \quad (1)$$

where U_u and V_i represent user-specific and item-specific latent feature vector respectively, N_u represent the friend set of user u , and $R_{u,i}$ represent the final prediction. Parameter α controls the effects of neighbors on the estimated rating. $T_{u,v}$ stands for trust value between user u and v . $g(\cdot)$ is a sigmoid function.

However, their work just care about user social friend relations. In our real life, even some of our friends may have totally different tastes in a social network. Maybe they are friends just because they are co-workers, partners, relatives, or maybe a celebrity who pays little attention to ordinary people.

We believe that users' social action information can provide more valuable clue to improve collaborative filtering performance. Our model works under an assumption that actions between users on the social network are conducted as they share very similar values and interests. The more frequently they interact, the higher similar their interests match. Incorporating with users' social action relations, we can magnify the affect factor of similarity to those who really share same interest and contact frequently while decrease the factor to those who are just normal friend and have little common interest.

Based on the STE model [10] Hao M. et al. proposed, we merge our social actions as a new factor into the model. The graphical model of our proposed approach is illustrated in Fig.1.

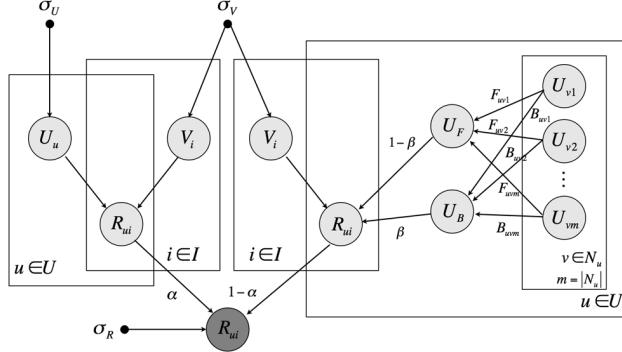


Fig. 1. The graphical model of social actions ensembled approach

In Fig.1, U_F and U_B represent user-friends and user-action latent feature vector respectively. The parameter α controls the final mixture of original PMF model and STE model and parameter β controls the mixture between social relations and actions. The weight $F_{u,v}$ is represented for the similarity between user u and user v , and the weight $B_{u,v}$ is represented for the intensity of user u 's and user v 's social actions.

As the assumption we just mentioned above, we define the user-action metrics as:

$$B_{u,v} = g(|N_{u,v}|) , \quad (2)$$

where $N_{u,v}$ is the frequency of actions between u and v . $g(\cdot)$ is a sigmoid function that map frequency into interval $[0,1]$.

According to probabilistic matrix factorization model, we redefine the new model conditional distribution over the observed ratings as

$$\begin{aligned} p(R|U, V, F, B, \sigma^2) = \\ \prod_{i=1}^N \prod_{j=1}^M [\mathcal{N}(R_{ij} | \sum_{k \in S(i)} (\beta B_{i,k} + (1 - \beta) F_{i,k}) U_k^T V_j, \sigma^2)]^{I^{ij}} , \end{aligned} \quad (3)$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and σ^2 , $S(i)$ stands for the neighborhoods of user i , and I^{ij} is the indicator function that is equal to 1 if user i rated movie j and equal to 0 otherwise. Hence, similar to Eq.3, through a Bayesian inference, we have

$$\begin{aligned} p(U, V|R, F, B, \sigma_F^2, \sigma_B^2, \sigma_U^2, \sigma_V^2) \propto \\ p(R|F, B, U, V, \sigma_F^2, \sigma_B^2) p(U|F, B, \sigma_U^2) p(V|F, B, \sigma_V^2) . \end{aligned} \quad (4)$$

4 Experiments

In this section, we conduct several experiments to compare the recommendation performance of our approach with other state-of-the-art recommendation methods.

4.1 Dataset Description

We choose Douban as the data source for our experiments on recommendation incorporating with social actions. We crawled movies' ratings, users and comments, and used these users as seed to further crawl their social network information. Finally, we obtain 244,015 unique users and 879 movies with 1,415,120 movie ratings. We got 8,394,452 record of user-friend relations with 5554 max degree and 10.2 average degree, 71,181 record of user-comment relations with 5355 max degree and 5.7 average degree. By the way, we also validate the dataset we crawled has a long-tail effect.

4.2 Experimental Results

In this section, in order to show the performance improvement of our social action approach, we compare our method with the following approaches:

1. **PMF**: this method is proposed by Salakhutdinov and Minh in [7]. It only uses user-item matrix for recommendations, and it is based on probabilistic matrix factorization.
2. **STE**: this method is proposed by Hao Ma in [10]. It uses social trust relations and mixes it with original PMF model by controlling with an α value.

We conduct our experiment with 80% training data and to predict the remaining 20% of ratings. The random selection was carried out 10 times independently. The curves of MAE and RMSE are shown in Fig.2(a).

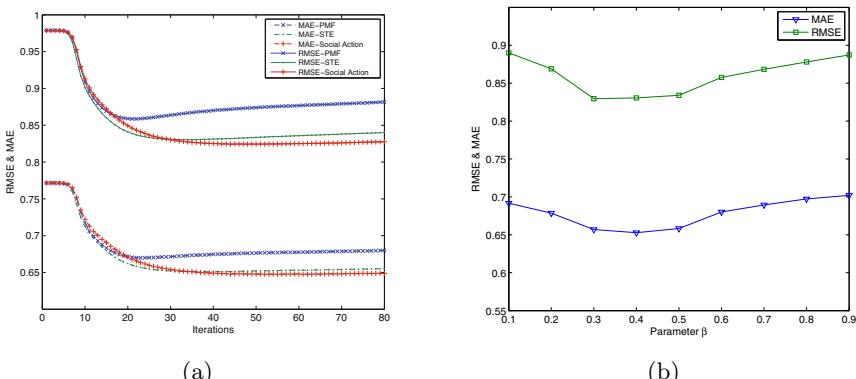


Fig. 2. The experimental results on Douban dataset

The experimental results² on the Douban dataset clearly demonstrate that social action incorporated model indeed outperforms existing methods for social network based recommendation. By incorporating more social information, we extend previous normal social friend relations. Hence, our model further decrease the noise of friend relations and improve the performance of recommender system.

5 Conclusions and Future Work

In this paper, we proposed a novel approach incorporating users social actions into a model-based approach for recommendation using probabilistic matrix factorization. Similar to the STE model presented in [10], our social action based model learns the latent feature vectors of users and items. Different from STE, we incorporate more social action information apart from social relation information to extend normal social friend relation. This extra information can reduce the bias and asymmetric relations. This work suggests several interesting directions for future work. Further, we want to incorporate more social action information not only just comments. Our model is a generic approach and can easily add other social action information into the model, such as forwards, mentions, and follows information.

Acknowledgments. This work is funded by the National Program on Key Basic Research Project (973 Program, Grant No.2013CB329605), Natural Science Foundation of China (NSFC, Grant Nos. 60873237 and 61003168), Natural Science Foundation of Beijing (Grant No.4092037), Outstanding Young Teacher Foundation and Basic Research Foundation of Beijing Institute of Technology, and partially supported by Beijing Key Discipline Program.

References

1. Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 114–121 (2005)
2. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Transactions on Information Systems, 89–115 (2004)
3. Salakhutdinov, R., Mnih, A.: Bayesian probabilistic matrix factorization using markov chain monto carlo. In: ICML 2008: Proceedings of the 25th International Conference on Machine Learning (2008)
4. Canny, J.: Collaborative filtering with privacy via factor analysis. In: SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 238–245 (2002)
5. Shen, Y., Jin, R.: Learning personal+social latent factor model for social recommendation. In: KDD 2012 (2012)

² We use 5 dimensions to represent the latent features, $\alpha = 0.4$ and $\beta = 0.4$, λ_U and λ_V are both equal to 0.04.

6. Rennie, J.D.M., Srebro, N.: Fast maximum margin matrix factorization for collaborative prediction. In: ICML 2005: Proceedings of the 22nd International Conference on Machine Learning (2005)
7. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, vol. 20 (2008)
8. Srebro, N., Jaakkola, T.: Weighted low-rank approximations. In: ICML 2003: Proceedings of the 20th International Conference on Machine Learning, pp. 720–727 (2003)
9. Massa, P., Avesani, P.: Trust-aware collaborative filtering for recommender systems. In: Meersman, R., Tari, Z. (eds.) CoopIS/DOA/ODBASE 2004. LNCS, vol. 3290, pp. 492–508. Springer, Heidelberg (2004)
10. Hao, M., Irwin, K., Lyu, M.R.: Learning to recommend with social trust ensemble. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 2009, p. 203 (2009)

DiffR-Tree: A Differentially Private Spatial Index for OLAP Query^{*}

Miao Wang¹, Xiaojian Zhang^{1,2}, and Xiaofeng Meng¹

¹ School of Information, Renmin University of China, Beijing, China

² School of Computer and Information Engineering,

Henan University of Economics and Law

{miaowang,xjzhang82,xfmeng}@ruc.edu.cn

Abstract. Differential privacy has emerged as one of the most promising privacy models for releasing the results of statistical queries on sensitive data, with strong privacy guarantees. Existing works on differential privacy mostly focus on simple aggregations such as counts. This paper investigates the spatial OLAP queries, which combines GIS and OLAP queries at the same time. We employ a differentially private R-tree(DiffR-Tree) to help spatial OLAP queries. In our method, several steps need to be carefully designed to equip the spatial data warehouse structure with differential privacy requirements. Our experiments results demonstrate the efficiency of our spatial OLAP query index structure and the accuracy of answering queries.

Keywords: Differential privacy, Spatial Index, OLAP.

1 Introduction

Spatial data exists pervasively in business information systems. It is claimed that 80% of the overall information stored in computers is geo-spatial related [1]. While, geographic Information Systems(GIS) and On Line Analytical Processing(OLAP) integrated queries(briefly called spatial OLAP queries) have been a rising type of queries in spatial data. Because the multidimensional spatial data in data warehouse does not have explicit or implicit concept hierarchy, traditional data warehouse and OLAP techniques cannot exploit spatial information in coordinates fully. This model of query is firstly implemented in 2007 by Leticia Gomez[2], it presents a formal model for representing spatial data, integrates in a natural way geographic data and information contained in data warehouses external to the GIS. This type of spatial aggregation is very common when the data analyzers do not want to concern about a specific person's data but the overall statistics of the data distribution. For example, “the average

* This research was partially supported by the grants from the Natural Science Foundation of China (No. 61070055, 91024032, 91124001); the National 863 High-tech Program (No. 2012AA010701, 2013AA013204); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University(No. 11XNL010).

salaries of 30-40 year-old males in District Haidian of Beijing, China". While, in many cases, the underlying spatial data contains sensitive information, such as the *disease* or *salary* in the OLAP information part, and the *location information* in the GIS information part. Then query processing on the sensitive data may compromise individual's privacy. Especially, based on the background knowledge, an adversary can infer sensitive information about a specific individual from spatial queries. For example, if an adversary knows that Mary is the only aged 30-40 female in a query region A, then he/she can get Mary's salary 50k by submitting "the average salary of 30-40 female in the region A" query. Another example, if the adversary possesses powerful background knowledge, he/she knows everyone's information in the Region A except Mary's, then he/she ask for "the average salary in Region A", and combines the query result and his own background knowledge to infer Mary's salary information.

To prevent such information leakage, the query results must be sanitized before release. Although different privacy models can be used, one of the only models that provide strong privacy guarantees is *differential privacy*[3]. The main idea of differential privacy is to inject sufficient noise to the query results so that an adversary cannot decide whether the information of a particular record owner was used to generate the sanitized result or not. In this paper, we utilize the notion of differential privacy[3] to support the spatial OLAP query index structure which combines data cubes and R-tree.

Contributions. First, We present an interactive differentially private framework to answer the GIS-OLAP integrated spatial OLAP queries for some low timeliness data sets. Second, we implement the framework by combining R-tree structure and OLAP technique with differential privacy. Based on this combination, a differentially private R-Tree(DiffR-Tree) index is built to support the OLAP queries (as we have shown in Section 4). Thirdly, we design heuristic query method satisfying differential privacy to further improve the efficiency of query processing. Finally, extensive experiments using real datasets demonstrate that our algorithm achieves highly accurate query results.

2 Related Work

Spatial OLAP system is designed to link the spatial data and the multidimensional data for OLAP queries in a specific spatial region bound. GIS-OLAP integration is firstly raised by Rivest et al.[5], it introduces the concept of SOLAP, which means standing for Spatial OLAP, and also talks about a SOLAP system's necessary operators and features. However, this work does not define a formal model for GIS-OLAP integration. Then Han et al.[6] present a model named Spatial Data cube, which adapts OLAP techniques to materialize spatial objects. But the model only supports spatial objects aggregation. After several former works, Rao et al.[7] and Zang et al.[8] attempt a novel approach to integrate GIS and OLAP for spatial data warehouses query processing, in this method they use R-tree to access data in fact tables. There are some other works but none of them present a formal model of integrating GIS and OLAP information until Gomez et al.[2] defines a formal

GIS and OLAP integrated model for spatial aggregation, and also formalizes a notion geometric aggregation.

While, when handling the query processing in spatial aggregation, the privacy of the data becomes a concern for researchers. And a lot of privacy protection algorithms have been proposed and promoted all the time, such as k-anonymity[9], l-diversity[10] and so on. While, Differential privacy is a new privacy-preserving model which is presented by C.Dwork[3]. It provides a strong privacy guarantee that the outcome of a calculation to be insensitive to any particular record in the data set. During the last five years, differential privacy has been adapted to a lot of applications. The most popular research is on differential private query processing, Hay et al.[11] proposes the adoption of count queries with a range selection on the attributes of a one-dimension database. Not only the counting queries, there are also a number of other specific types of queries have been adapted to differential privacy, for example, graph queries[12], histogram analysis queries[11]and linear counting queries[13]. Xiao et al.[14][15] proposes wavelet transforms, which is to adapt differential privacy to multi-dimension data count queries and achieves the same utility as Hay et al.s solution. Cormode et al.[16] propose several spatial data decomposition techniques under differential privacy, which are based on quadtree and kd-tree. The two type of decompositions first partition the spatial data and then add the noise the transformed data. Moreover, Friedman and Schuster[17] construct a differentially private decision tree to do differentially private knowledge discovery, and Bhaskar et al.[18] present an approach to solve the frequent pattern mining problems in a differentially private way. In contrast to the above research works on specific analysis purposes, several differential privacy systems have been implemented to operate on a wide variety of data queries. Specially, PINQ[19] and GUPT[20]. PINQ[19] does not consider the probability that the application developer may be an adversary, it enables application developers or data analyzers to write differential private programs using basic functional blocks of differential privacy, such as exponential mechanism, Laplace mechanism, etc. While GUPT[20] provides a flexible system that supports many different types of data analysis programs, it achieves reasonable accuracy for problems like clustering and regression. Ding et al.[21] introduce a general noise-control framework to release all or part of a data cube in an ϵ -differentially private way. The paper only focuses on reducing the noise in the released cuboids.

3 Preliminaries

3.1 Differential Privacy

Differential privacy, in general, guarantees that changing or removing any record from a database has negligible impact on the output of any analysis based on the databases. Therefore, an adversary will learn nothing about an individual, regardless of whether her record is present or absent in the database. Formally, in the context of incremental updates, differential privacy is defined below.

Definition 1. (*Neighboring Datasets*) We say two datasets D and D' are neighboring datasets when they contain one record different, that is $|D \Delta D'|=1$.

Definition 2. (ϵ -differential privacy) A randomized algorithm Ag is differentially private if for all datasets D and D' where they are neighbor datasets, and for all possible anonymized datasets \hat{D}

$$\Pr[Ag(D) = \hat{D}] \leq e^\epsilon \times \Pr[Ag(D') = \hat{D}], \quad (1)$$

where the probabilities are over the randomness of the Ag . The parameter $\epsilon > 0$ controls the level of privacy.

To introduce the mechanism to satisfy differential privacy, it is necessary to understand *Sensitivity*. The sensitivity of an algorithm expresses the maximal possible change of its outputs from two neighbor datasets.

Definition 3. (*Sensitivity*) For any function $f: D \rightarrow \mathbb{R}^d$, the sensitivity of f is

$$\Delta f = \max_{D, D'} \|f(D) - f(D')\|_1 \quad (2)$$

for all neighbor datasets D, D' .

Given the sensitivity of a function f , there are two popular mechanisms to achieve ϵ -differential privacy.

Laplace Mechanism. Dwork et al.[22] propose the Laplace mechanism. The mechanism takes a dataset D , a function f , and a parameter λ that determines the magnitude of noise as inputs. It first computes the true output $f(D)$, and then perturbs the output by adding laplace noise. The noise is generated according to the Laplace distribution with probability density function $Pr(x|\lambda) = \frac{1}{2\lambda} \exp(-\frac{|x|}{\lambda})$. The following theorem connects the sensitivity with the magnitude of noise, and guarantees the perturbed output $f(\hat{D}) = f(D) + Lap(\lambda)$ satisfying ϵ -differential privacy, where $Lap(\lambda)$ is a random variable sampled from the Laplace distribution.

Theorem 1. For any function $f: D \rightarrow \mathbb{R}^d$, the algorithm Ag that adds independently generated noise with distribution $Lap(\Delta f/\epsilon)$ to each of the d outputs satisfies ϵ -differential privacy.[22]

Exponential Mechanism. Mcsherry and Talwar[23] propose the exponential mechanism that can choose an output $t \in T$ that is close to the optimum with respect to a utility function while preserving differential privacy. The exponential mechanism takes as inputs a data set D , output range T , privacy parameter ϵ , and a utility function $u: (D \times T) \rightarrow \mathbb{R}$ that assigns a real valued score to every output $t \in T$, where a higher score means better utility. The mechanism induces a probability distribution over the range T and then samples an output t from the whole range T . Let $\Delta u = \max_{D, D'} |u(D, t) - u(D', t)|$ be the sensitivity of the utility function. The probability associated with each output is proportional to $\exp(\frac{\epsilon u(D, t)}{2\Delta u})$; that is, the output with a higher score is exponentially more likely to be chosen.

Theorem 2. For any function $u : (D \times T) \rightarrow \mathbb{R}$, an algorithm Ag that chooses an output t with probability proportional to $\exp(\frac{\epsilon u(D, t)}{2\Delta u})$ satisfies ϵ -differential privacy.[23]

Besides, there are two important composition properties of differential privacy, which is necessary for the privacy budget ϵ allocation on our method.

Lemma 1. (Sequential Composition[19]) Let each randomized algorithm Ag_i provides ϵ_i -differential privacy. A sequence of $Ag_i(D)$ over the data set D provides $(\sum_i \epsilon_i)$ -differential privacy.

Lemma 2. (Parallel Composition[19]) Let each randomized algorithm Ag_i provides ϵ_i -differential privacy. A sequence of $Ag_i(D_i)$ over a set of disjoint data sets D_i provides $(\max_i \epsilon_i)$ -differential privacy.

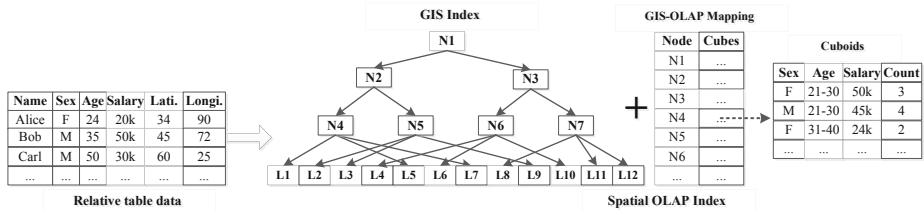
4 Algorithm

4.1 The Framework

The framework of our spatial OLAP query index is shown in Fig 1. The framework contains four steps as shown in Algorithm 1. Firstly we read the record data from a data file which contains the OLAP information and the location GIS information at the same time. Secondly, we build the R-tree using the location information of records, which is the GIS index of our final query index DiffR-tree. Then we transform the data records located in the MBR of each leaf into a data cube, and put the key-value pair (leafID, cube) into the OLAP data container of DiffR-tree. Thirdly, to improve the query efficiency, we pre-computation the data cubes of each non-leaf node in R-tree. After that, to protect the data sensitive information, we add Laplace noise to each data cube. But the laplace noises would make the metrics in data cubes be deprived of consistency, so we have to refine the data to keep data consistency in the spatial OLAP query index. At last, we have built a differentially private spatial OLAP index, we design an adaptive heuristic process based on Exponential mechanism to improve the query efficiency further, which satisfies differential privacy and keeps the accuracy as high as possible. We introduce the key steps in the following several subsections.

4.2 The Construction of DiffR-tree

The method to building the differentially private spatial OLAP index is described in Algorithm 2. The DiffR-tree has two main parts: the GIS part and the OLAP part. We use R-tree as our GIS part, and the data is the location information of records in the database table(Line 2). While to build the OLAP part, we need to transform the database data into data cubes. In this step, we transform the records located in each MBR of leaf nodes into data cube, and link them into DiffR-tree OLAP parts(Line 3-5). The transform algorithm is very similar to the attribute-oriented induction method in common OLAP process(Line 6-10), we do not talk about it in detail here.

**Fig. 1.** The framework of DiffR-Tree**Algorithm 1. DiffR-tree**

Input: D : a data set with GIS and OLAP information; ϵ : privacy budget; Q : a spatial OLAP aggregation query

Output: The differentially private query result;

- 1: **Read Data:** Read the GIS and OLAP information which are stored in respective columns in D ;
- 2: **Build DiffR-tree:** Use the GIS information data in T to build a R-tree. Then induce the OLAP information into data cube with Laplace noise, and link the data cubes to leaf nodes in R-tree. This step is shown in Algorithm 2;
- 3: **Pre-Computation and Enforce Consistency:** Link a Laplace noisy data cube to each non-leaf node in the R-tree in a bottom-top way, then keep the OLAP data consistency in a top-down way. This step is shown in Algorithm 3;
- 4: **Process Query:** Query processing in a top-down way, and in each non-leaf node, use Exponential mechanism proportional to decide whether the node's subtree to be further explored or use the node's linked data cube to contribute to query result. This step is shown in Algorithm 4;
- 5: **Result:** Integrate the returned information and then output the final noisy query result.

Algorithm 2. BDiffR-tree (T, ϵ)

- 1: $T \leftarrow \emptyset$; // T refers to the DiffR-tree
- 2: Build R-tree $tree$ using record location as points, $T.GIS_index = tree$;
- 3: **for** leaf _{i} in $T.leafnodes$ **do**
- 4: leaf _{i} .cube \leftarrow TransformToCube(records in leaf _{i});
- 5: $T.OLAP_cubes \leftarrow T.OLAP_cubes + leaf_i.cube$;
- 6: **TransformToCube(r)** // r refers to the records in some leaf
- 7: Dimensions $\xrightarrow{}$ query related dimensions and their respective segments;
- 8: Create cuboids by combining every segments in all dimensions to be a data cube;
- 9: **for** each record i ($i \in [1, r]$)**do**
- 10: **for** each metric j ($j \in [1, m]$) **do**
- 11: metric _{j} \leftarrow metric _{j} + record _{i} .data _{j} ;

4.3 Pre-computation and Enforce Consistency

In the above step, we just link the leaf nodes to data cubes. To improve the query efficiency in the following query step, we also need to get the non-leaf nodes linked with data cubes. Here we use a bottom-top way to compute a non-leaf node related data cube by combining the data cubes of its subnodes. After finishing this step, all of the nodes in the DiffR-tree GIS index has corresponding data cubes in DiffR-tree OLAP parts(Line 2-6).

But we need to realize that all of data cubes have not been noised, since R-tree and data cubes are both data-dependent structure, if we use the index to process queries directly the sensitive data in original records will be leaked to the adversary. So we need to add noise in a differentially private method to protect the privacy in the DiffR-tree(Line 7-8). Because of the Sequential Composition and Parallel Composition properties introduced in Section 3, we can get that in the GIS part, a path from root to leaf conforms to Sequential Composition property, and the nodes in the same layer are disjoint. So if the privacy budget is ϵ , then each node should be allocated $\frac{\epsilon}{h}$, here h is the height of the tree. In the OLAP part, the allocated $\frac{\epsilon}{h}$ privacy budget of each node would be used to add Laplace noise into data cubes. Since every metric in a data cube is independent, so each one metric can get $\frac{\epsilon}{h}$ budget.

Since Laplace noise is random, so it may occur data inconsistency. For example, if we measure *count* metric, some node related cubes it is 7, but the its subnodes counts are 2,6,1, the sum value is inconsistent with 7. So to enforce consistency we need to refine the metrics number in a top-down way(Line 9-14), which is shown in Algorithm 3.

Algorithm 3. PCEC(T, ϵ)

Pre-Computation(T, ϵ)

```

1: for  $h=T.\text{height}-1, \dots, 1, 0$  do
2:    $n_h \leftarrow$  the nodes number of level  $h$ ;
3:   for each  $i(i \in [1, n_h])$  do
4:     if  $\text{node}_i.\text{type}$  is NON\_LEAF then
5:       for each  $j(j \in [1, c])$  in  $\text{node}_i.\text{cube}.\text{cuboids}$  do
           $\text{cuboid}_j \leftarrow \text{cuboid}_j + \sum \text{node}_i.\text{subNode}.\text{cuboid}_j$ ;
           $T.\text{OLAP\_cubes} \leftarrow T.\text{OLAP\_cubes} + \text{node}_i.\text{cube}$ ;
6:     else if  $\text{node}_i.\text{type}$  is LEAF then return;
7:   for cube in DiffR-tree.OLAP_cubes do
8:     for each  $i(i \in [1, m])$  in cube.metrics do
         $\text{metric}_i \leftarrow \text{metric}_i + \text{LaplaceNoise}(\frac{3*h}{\epsilon})$ ;
9: Keep Consistency(metrics)
10:  for  $h=0, 1, \dots, T.\text{height}-2$  do
11:     $n_h \leftarrow$  the nodes number of level  $h$ ;
12:    for each  $i(i \in [1, n_h])$  do
13:      for node in  $\text{node}_i.\text{subnodes}$  do
14:         $\text{node}.\text{metric}_j \leftarrow \text{node}_i.\text{metric}_j * \frac{\text{node}.\text{metric}_j}{\sum \text{node}.\text{metric}_j}$ ;
```

4.4 Query Processing

After building a differentially private query index, we can use it to process spatial OLAP aggregation queries, the algorithm is shown in Algorithm 4. In this step we have two important techniques. The first one is the adaptive heuristic selection based on Exponential mechanism. In a traditional heuristic in R-tree query processing, if the joint area percentage between the query area and the non-leaf node MBR area is larger than a threshold, then return the non-leaf node information. In our method, we modify the percentage threshold into a probability selection event. We use the Exponential selection directly proportional to the joint area. We define $u(D, t)$ as the joint area percentage, and the output range T is {TRUE,FALSE} which represent whether to traverse sub nodes. That is, the larger the joint area, the bigger the probability to use the non-leaf node as part of the query result(the choice is TRUE), or it goes down to traversal sub nodes(Line 8-15)(the choice is FALSE). With the resulted elements, we integrate the data information and get the query result to return to query analyzers.

Algorithm 4. Process Query(T, ϵ)

```

1: query←Parse query file; root←DiffR-tree.GIS_index.rootID;
2: queryNodes ← GetQueryReturnedNodes( $T.\text{root}$ ,query);
3: for node in queryNodes do
4:   if node.type is Record then add record.data+LaplaceNoise( $\frac{3}{\epsilon}$ ) to result;
5:   else if node.type is LEAF or node.type is NON_LEAF
6:     add joint percentage of node.cube data to result;
7:   Deal with the data and return query result;
GetQueryReturnedNodes( $T.\text{node}$ ,query)
8: queryNodes←  $\emptyset$ ;
9: for each  $i(i \in [1,n])$  in node.subnodes do
10:   jointArea←JoinArea(subnode $_i$ .MBR, query.range);
11:   jointPercentage← $\frac{\text{jointArea}}{\text{subnode}_i.\text{MBR.area}}$ ;
12:   choose←ExponentialSelection $\propto$  jointPrecentage;
13:   if choose is TRUE then queryNodes.add(subnode $_i$ );
14:   else queryNodes.addAll(GetQueryReturnedNodes(subnode $_i$ ,query));
15: return queryNodes;
```

5 Privacy Analysis

In this section, we use composition properties to prove that our algorithm satisfies differential privacy. We know that any sequence of computations that each provides differential privacy in isolation is differentially private in sequence, and any parallel computations can also satisfy differential privacy in parallel. Then we can get the following theorem.

Theorem 3. *Algorithm 1 is ϵ -differential private.*

Proof. Our algorithm can be partition into two parts: build index and query processing. First of all, we allocate $\frac{\epsilon}{3}$ privacy budget to the original record data to make it satisfy $\frac{\epsilon}{3}$ -differential privacy. Then in the index tree, we allocate $\frac{\epsilon}{3*h}$ privacy budget to each node in the R-tree. Since nodes in the same layer satisfy Parallel Composition property and a path from root to leaf satisfies Sequential Composition property, then we know the DiffR-tree building satisfy $\frac{\epsilon}{3*h} * h = \frac{\epsilon}{3}$ -differential privacy. Last in the query processing, since which layer the query goes to and what data cubes the query returns are data-dependent, so it is necessary to protect the query process privacy. Here we use Exponential selection adapted heuristic algorithm, and allocate $\frac{\epsilon}{3}$ privacy budget for Exponential mechanism. According to Lemma 1, Algorithm 1 satisfies $\frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon$ -differential privacy.

6 Experiments

To evaluate the accuracy and efficiency of our method, we perform the experiments on synthetical and real data sets. All of these data sets have the following necessary information: *sex*, *age*, *salary*, *latitudeLocation*, *longitudeLocation*. The two synthetical data sets are generated following normal and uniform distribution respectively. The real data set is the location-based data set from Stanford Large Network Dataset Collection[24]. The data sets are depicted in Figure 3(uniformly distributed data), Figure 4(normally distributed data), Figure 5(real data).

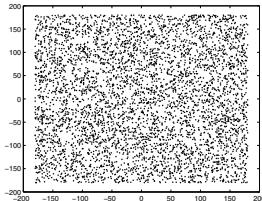


Fig. 2. Uniformly Dist. Data

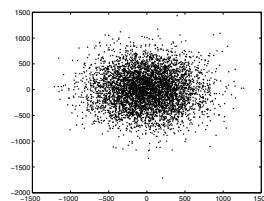


Fig. 3. Normally Dist. Data

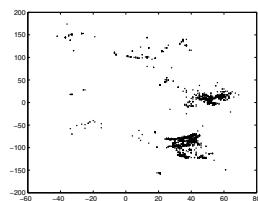


Fig. 4. Real Data

We will evaluate the query processing accuracy when considering two factors: query size and total privacy budget ϵ . We change the query size from 10% to 90% of the whole data area, the privacy budget from 0.1 to 1. Given a query size s and a privacy budget ϵ , we generate 100 queries of the given query size with query region's center randomly generated, and build index then process query in ϵ -differential privacy, we measure the average accuracy of the 100 queries results.

Moreover, to prove the efficiency of our algorithm, we compare the query accuracy and efficiency with a baseline. In the baseline, we traversal the index tree until the records in the bottom layer, and return the query results without adding noises.

From the experiment results from Figure 5,6,7, we can get the relative error rate by changing the query size from 10% to 90% in uniformly distributed data, normally distributed data and real data respectively, here we set $\epsilon = 1$. We can get one common trend from the three figures that when query size increases, the relative error rate decreases. Another trend is when the original data size increases, the relative error decreases. The reason for both the trends is that the more points the query region contains, the heuristic technique and estimation in query processing are more accurate.

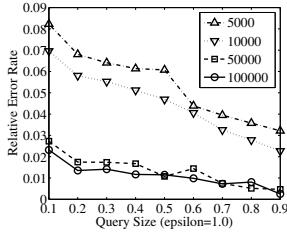


Fig. 5. Uniformly Dist. Data

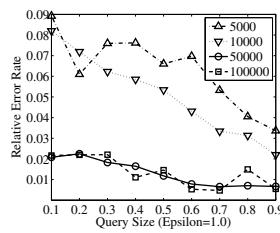


Fig. 6. Normally Dist. Data

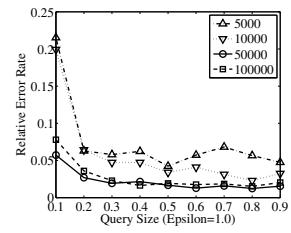


Fig. 7. Real Data

The Figure 8,9,10 show the relative error rate when changing privacy budget from 0.2 to 1.0 in uniformly distributed data, normally distributed data and real data respectively, here we set querySize=50%. We can know from the experimental results in three figures that when the privacy budget ϵ increases, the accuracy increases obviously. That means, we should choose $\epsilon = 1$ if we want to get the best accuracy.

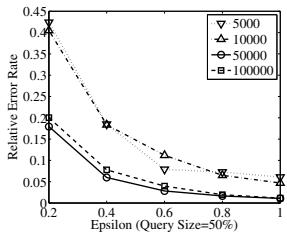


Fig. 8. Uniformly Dist. Data

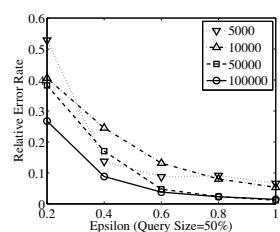


Fig. 9. Normally Dist. Data

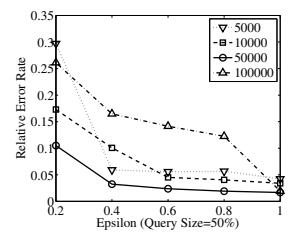


Fig. 10. Real Data

In Figure 11, we prove our method is more efficient the R-tree query processing, in this figure, we depict the time cost (in μ s) between DiffR-tree and R-tree query performance in uniformly distributed data of size 10000, 50000 and 100000. We can see that when the original data size increases, the primordial R-tree query processing time increases sharply, while the time cost of our method holds steady and even decreases. When data size is larger, the difference in time

cost is obvious. This is because we use differentially private adaptive heuristic technique to handle queries and return the non-leaf as results. Since the above experiment results show the accuracy of our method, the experiment indicates that the perfect query efficiency of our method.

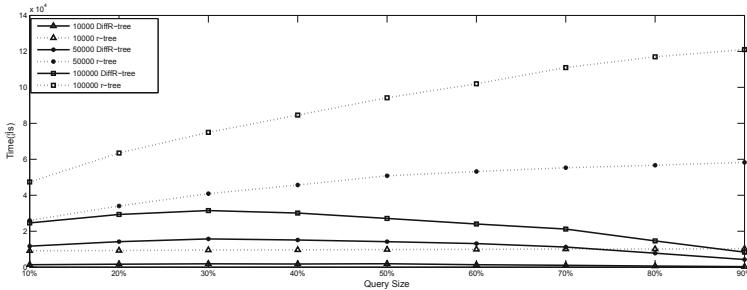


Fig. 11. Time Cost Experiments

7 Conclusion

In this paper, we propose a differentially private spatial OLAP query index and implement it. We use R-tree as the GIS index of our spatial OLAP query index, then combine data cube technique as OLAP part with the spatial index, while all of the algorithm satisfies differential privacy. In our method, we design the key steps of building GIS-OLAP integrated index, enforcing consistency, and heuristically handling queries in a differentially private way, and we prove it satisfy differential privacy in a theorematic proof. Experiments results show the accuracy and efficiency of our algorithm.

References

1. Daratech: Geographic Information Systems Markets and Opportunities. Daratech, Inc. (2000)
2. Gómez, L., Haesvoets, S., Kuijpers, B., Vaisman, A.: Spatial aggregation: Data model and implementation, CoRR abs/0707.4304 (2007)
3. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
4. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publisher (2001)
5. Rivest, S., Bédard, Y., Proulx, M.-J., Nadeau, M., Hubert, F., Pastor, J.: SOLAP: Merging Business Intelligence with Geospatial Technology for Interactive Spatio-Temporal Exploration and Analysis of Data. Journal of International Society for Photogrammetry and Remote Sensing 60(1), 17–33 (2005)
6. Han, J., Stefanovic, N., Koperski, K.: Selective materialization: An efficient method for spatial data cube construction. In: Wu, X., Kotagiri, R., Korb, K.B. (eds.) PAKDD 1998. LNCS, vol. 1394, pp. 144–158. Springer, Heidelberg (1998)

7. Rao, F., Zang, L., Yu, X., Li, Y., Chen, Y.: Spatial hierarchy and OLAP-favored search in spatial data warehouse. In: Proceedings of DOLAP 2003, Louisiana, USA, pp. 48–55 (2003)
8. Zhang, L., Li, Y., Rao, F., Yu, X., Chen, Y.: An approach to enabling spatial OLAP by aggregating on spatial hierarchy. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2003. LNCS, vol. 2737, pp. 35–44. Springer, Heidelberg (2003)
9. Sweeney, L.: k-anonymity: A Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 557–570 (2002)
10. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkatasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery From Data, TKDD (2007)
11. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the Accuracy of Differentially Private Histograms through Consistency. Proceedings of the VLDB Endowment, 1021–1032 (2010)
12. Karwa, V., Raskhodnikova, S., Smith, A., Yaroslatsky, G.: Private analysis of graph structure. In: VLDB (2011)
13. Li, C., Hay, M., Rastogi, V., Miklau, G., McGregor, A.: Optimizing linear counting queries under differential privacy. In: PODS (2010)
14. Xiao, X., Wang, G., Gehrke, J.: Differential Privacy via Wavelet Transforms. In: ICDE, pp. 225–236 (2010)
15. Xiao, X., Wang, G., Gehrke, J.: Differential Privacy via Wavelet Transforms. IEEE Transactions on Knowledge and Data Engineering, 1200–1214 (2011)
16. Cormode, G., Procopiuc, M., Shen, E., Srivastava, D., Yu, T.: Differentially Private Spatial Decompositions. In: ICDE (2012)
17. Friedman, A., Schuster, A.: Data Mining with Differential Privacy. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA, pp. 493–502 (2010)
18. Bhaskar, R., Laxman, S., Smith, A., Thakurta, A.: Discovering Frequent Patterns in Sensitive Data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA, pp. 503–512 (2010)
19. McSherry, F.: Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis. In: SIGMOD (2009)
20. Mohan, P., Thakurta, A., Shi, E., Song, D., Culler, D.E.: Gupt: Privacy Preserving Data Analysis Made Easy. In: SIGMOD, Scottsdale, Arizona, USA, May 20-24 (2012)
21. Ding, B., Winslett, M., Han, J., Li, Z.: Differentially Private Data Cubes: Optimizing Noise Sources and Consistency. In: SIGMOD, Athens, Greece (2011)
22. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
23. McSherry, F., Talwar, K.: Mechanism Design via Differential Privacy. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 94–103 (2007)
24. Location-based online social networks data in Stanford Large Network Dataset Collection, <http://snap.stanford.edu/data/index.html#locnet>

K-core-preferred Attack to the Internet: Is It More Malicious Than Degree Attack?

Jichang Zhao¹, Junjie Wu², Mingming Chen¹, Zhiwen Fang¹, and Ke Xu¹

¹ State Key Laboratory of Software Development Environment, Beihang University

² Beijing Key Laboratory of Emergency Support Simulation Technologies for City Operations, School of Economics & Management, Beihang University

Abstract. K-core (k-shell) index is an interesting measure that describes the core and fringe nodes in a complex network. Recent studies have revealed that some high k-core value nodes may play a vital role in information diffusion. As a result, one may expect that attacking high k-core nodes preferentially can collapse the Internet easily. To our surprise, however, the experiments on two Internet AS-level topologies show that: Although a k-core-preferred attack is feasible in reality, it turns out to be less effective than a classic degree-preferred attack. Indeed, as indicated by the measure: normalized susceptibility, we need to remove 2% to 3% more nodes in a k-core-preferred attack to make the network collapsed. Further investigation on the nodes in a same shell discloses that these nodes often have degrees varied drastically, among which there are nodes with high k-core values but low degrees. These nodes cannot contribute many link deletions in an early stage of a k-core-preferred attack, and therefore make it less malicious than a degree-preferred attack.

Keywords: Robustness, K-core, AS-level Internet, Malicious Attack.

1 Introduction

The Internet has become the most important communication infrastructure in the world, especially after the explosion of online social networking sites. Tremendous research efforts have been devoted to scale-free networks, such as the AS-level Internet in the level of autonomous system [20,5,21,2]. Among them, attack survivability remains one of the core topics. People find that, while the Internet is robust to the random failure, it is fragile to malicious attacks, which are generally defined as removing important nodes or links from the networks preferentially [1]. Specifically, the simple degree-preferred attack, i.e., attacking the nodes with high degrees preferentially, is often regarded as the most probable attack type in reality. Other types of attacks, e.g., attacking the nodes with higher betweenness preferentially, may be more malicious than the degree-preferred attack, but often need the global topological information of networks and consume much more computational time [3], and thus become infeasible in practice [12].

K-core (k-shell) index is an interesting measure that categorizes the nodes in a complex network into the core nodes and the fringe ones. Recently, in their landmark paper [14], the authors found in many types of complex networks that k-core

value is a more effective measure to describe the influence of a node to the propagation of information or diseases. Indeed, they disclosed a surprising fact that some nodes with high degrees play a trivial role in the information spreading process. They argued that a k -core viewpoint is more instructive; that is, those high-degree nodes actually have low k -core values and thus locate in the fringe of the network. From this aspect, one may expect that a k -core-preferred attack, i.e., attacking high k -core nodes preferentially, can collapse the Internet more easily than a degree-preferred attack. This motivates our study on the k -core-preferred attack, which to our best knowledge is among the first few studies along this line.

To this end, we performed comparative experiments for the two types of malicious attacks on two real-world AS-level Internet data sets. Six measures including both the structural and propagative ones were introduced to characterize the damages to the networks during the attacks. To our surprise, the results show that: Although a k -core-preferred attack is feasible using the traceroute tool [9,13], it is less malicious than a classic degree-preferred attack. Indeed, as indicated by the normalized susceptibility measure, we need to remove 2% to 3% more nodes in a k -core-preferred attack to make the network collapsed. Further investigation on the nodes in a same shell disclosed that these nodes often have degrees varied drastically, among which there are nodes with high k -core values but low degrees. These nodes cannot contribute sufficient link deletions in an early stage of a k -core-preferred attack, and therefore make it less malicious than a degree-preferred attack.

2 Related Work

Weak attack survivability but strong error tolerance [1] is a dilemma for the complex networks. In recent years, many researchers focus on the robustness analysis and enhancement of complex networks. For instance, Cohen et al. unveiled that the Internet is resilient to random failures [6] but fragile to the intentional attack [7]. Holme et al. proposed four different attacking strategies and found that attacks by recalculating degrees and betweenness centrality are often more harmful than attacks based on the initial network [12]. Meanwhile, as a key metric in complex networks, k -core index also attracts a lot of research interests in the scope of the Internet. For example, Carmi et al. used information on the connectivity of network shells to separate the AS-level Internet into three subcomponents [5]. Zhang et al. found that the k -core with larger k is nearly stable over time for the real AS-level Internet [21]. Zhang et al. proposed a model based on k -core decomposition to model the Internet Router-level topology [22]. In the inspirational work [14], Kitsak et al. focused on evaluating the influence of a node in the spread of information or diseases through its k -core index. They reported an unexpected finding that some hub nodes may locate in the periphery of the network.

Despite of the existed abundant researches on the network robustness and k -core index, little work has been done to unveil whether the attack based on k -core is more malicious than other types of attacks to the AS-level Internet. This indeed motivates our study in this paper.

3 Preliminaries

In this section, we first discuss the feasibility of attacking the AS-level Internet, and then revisit the measures employed to characterize the damages of networks caused by malicious attacks. Finally, the real-world data sets employed in this paper are presented.

3.1 Feasibility of Attacking an AS in the Internet

The network of AS-level Internet stands for business relationships between different *Internet Service Providers (ISP)*. The recent survey by Kevin Bulter et al. revisited several attacking methods [4]. For instance, *prefix hijacking* means an AS A can advertise a prefix from address space belonging to another AS B ; then the traffic that should be routed to B would be routed to A falsely, which means AS B is deleted from the network. For another, *link cutting attack* can be manifested by either physically attacking a link or employing Denial-of-Service(DoS) attacks. In addition, there have been quite a few real-world AS-attacking cases in the history of the Internet [4]. For example, in 1997, a misconfigured router maintained by a small *ISP* in Florida injected incorrect routing information into the global Internet and claimed to have optimal connectivity to all Internet destinations. As a result, most Internet traffic was routed to this *ISP*, which overwhelmed the misconfigured router and crippled the Internet for nearly two hours [4]. To sum up, attacking an AS in the real-world Internet is indeed feasible. As a result, discussing attack survivability of the Internet in the level of AS makes physical sense.

3.2 K-Core index

The Internet can be intuitively modeled as a *graph* $G(V, E)$ at different levels, where V is the set of interfaces, routers or ASes, while E is the set of links between them. In this paper, we mainly focus on the AS-level Internet, which means a node stands for an AS and a link stands for the business relationship between its two ends. The number of links of a node is defined as its *degree* k . K-core [19] in a graph G could be defined as the maximum subgraph G^k , in which each node's degree is at least k . By recursively pruning the least connected nodes, the hierarchical structure of the network can be broken down to the highly connected central part, which is stated as the *core* of the network [10]. Then *k-core* (k-shell) index, denoted as k_s , is used to characterize how far a node is from the core of a network. A node i has k-core index k_s if it is in the k_s -core but not in the $(k_s + 1)$ -core. A larger k_s indicates the node is closer to the core. K-core can be computed through the following steps [5,14]. First, remove all the nodes with degree $k = 1$. After this step, there may appear new nodes with $k = 1$. Then keep on pruning these nodes until all nodes with degree $k = 1$ are removed. k_s of the removed nodes is then set to 1. Next, we repeat the pruning process in a similar way for the nodes with degree $k = 2$ and subsequently for higher values of k until all nodes are removed. After this process, the k-core values of all the nodes can be determined.

3.3 Measures of Network Robustness

We employ four structural measures to characterize the damage of a network. The *relative size of the giant connected component (GCC)*, denoted as f_{GCC} , is a generally used metric to quantify the extent to which a network is damaged. Another intuitive measure is the *number of disconnected clusters* in the network. The greater the number is, the more disconnected sub-networks are due to the attack, which indicates a more serious damage. We can normalize the number by dividing it by the size of the network, denoted as $f_{cluster}$. *Network efficiency* [16] is the only topology property we adopt in this paper, which relates strongly to global shortest paths. It is defined as

$$\Lambda = \frac{1}{N(N-1)} \sum_{i=1, j=1, i \neq j}^N \frac{1}{d_{ij}}, \quad (1)$$

where N is the size of the network and d_{ij} is the length of the shortest path between nodes i and j . A lower Λ means the averaged length of shortest paths in the network is longer and the network efficiency is lower. We also employ the *normalized susceptibility* [15], which is denoted as

$$\bar{S} = \frac{\sum n_s s^2}{N}, \quad (2)$$

where n_s is the number of components of size s . If there exists a phase transition in the variation of \bar{S} , it means that the network is already collapsed. However, the network is just shrinking if there is no phase transition during the attack.

In [8,17], the AS-level topology of the Internet was employed as the underlying network for worm spread investigation. Hence, we also adopt two propagative measures, corresponding to the Susceptible-Infected-Susceptible (SIS) model and the Susceptible-Infected-Recovered (SIR) model, respectively, to describe the damage status of an AS-level network. For the SIS model, nodes in the network are classified into two categories: the infected ones and the susceptible ones. Each susceptible node can be infected by its infected neighbors with a probability μ , meanwhile an infected one may return to the susceptible status with a probability β . As a result, we denote a SIS model as $SIS(\mu, \beta)$. As time evolves, the *fraction of the infected population* will eventually stabilize at a certain level, denoted as f_c^{SIS} . f_c^{SIS} can be used to characterize how far the disease can spread in the network, and thus reflect the damage status of the underlying network. All other things being equal, a smaller f_c^{SIS} implies a more severe damage. In the SIR model, a node in the network is in one of the three statuses: susceptible, infected and recovered. For a susceptible node, it may get infected by its infected neighbors with a probability μ , and an infected node may get recovered with a probability λ and will never be infected again. As a result, we denote the SIR model as $SIR(\mu, \lambda)$. Here, we utilize the *maximum fraction of nodes that get infected during the spreading process*, denoted as f_{max}^{SIR} , to characterize the worst situation.

Table 1. Traceroute samples

T_{id}	1	2	3	4	5
DIMES-AS	$T(1, 500)$	$T(26, 260)$	$T(53, 530)$	$T(132, 1320)$	$T(264, 2640)$
UCLA-AS	$T(1, 500)$	$T(38, 380)$	$T(76, 760)$	$T(191, 1910)$	$T(382, 3820)$
T_{id}	6	7	8	9	
DIMES-AS	$T(528, 5280)$	$T(793, 7930)$	$T(1057, 10570)$	$T(1321, 13210)$	
UCLA-AS	$T(764, 7640)$	$T(1146, 11460)$	$T(1528, 15280)$	$T(1910, 19100)$	

3.4 Real-World Network Topologies

It is hard to obtain an accurate and complete picture of the AS-level Internet. In order to make our results more reliable and convincing, we use two AS-level Internet data sets. The first one, denoted as **DIMES-AS**, comes from the project of DIMES¹. **DIMES-AS** was released in March, 2010 with 26424 nodes and 90267 links. The second data set, denoted as **UCLA-AS**, was released by Internet Research Lab in UCLA² in November, 2010. We extract the topology only from the map file released on Nov. 23, 2010 and get a network of 38200 nodes and 140726 links.

4 Modeling Attacks to the AS-Level Internet

In the section, we first give the definitions of attacks based on the degree and k-core values of network nodes, respectively. Then we demonstrate how to estimate the k-core index, which enables the k-core preferred attack to real-world networks.

4.1 Defining Attacks

Here we focus on two kinds of malicious attacks in the AS-level Internet. One is the attack based on the node degree, called *degree-preferred attack* (DA). The other is the attack based on the k-core index, called *k-core-preferred attack* (CA). In a degree-preferred attack, we sort all the nodes in the descending order of degrees and remove from the network the ones with high degrees first. Similarly, in a k-core-preferred attack, all the nodes in the network are ranked in the decreasing order of k-core values. Nodes located in the same shell, i.e., having a same k-core value, are further sorted in the decreasing order of degrees. Then the nodes will be removed from the highest rank to the lowest rank gradually. Note that we do not recalculate the nodes' degrees or k-core values after each wave of attack, as done in [12,18].

¹ <http://www.netdimes.org>

² <http://irl.cs.ucla.edu/topology/>

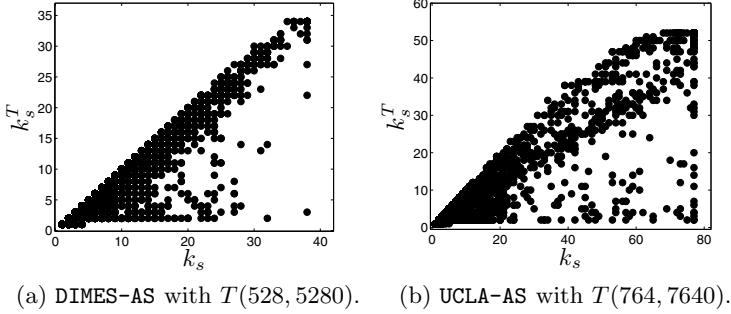


Fig. 1. Correlation between k_s^T and k_s

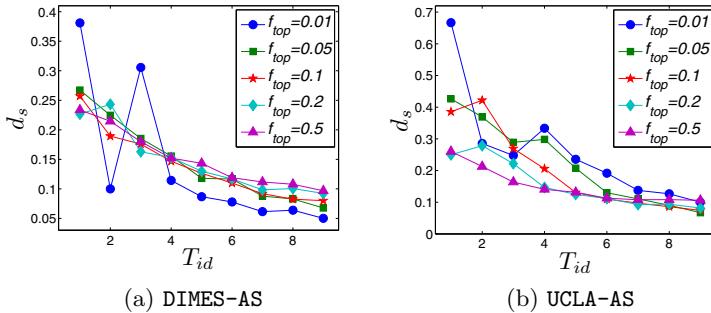


Fig. 2. Distances between sample sequences and real sequences

4.2 Estimating the K-Core Index

Generally speaking, the k-core index of a node is robust, i.e., it can be estimated from limited information of the network. To illustrate this, we perform simulations of traceroute [11] on the two AS-level topologies.

In the simulation, we randomly choose the sources and destinations from the network. Each simulation is denoted as $T(s, d)$, where s is the number of sources and d is the number of destinations. For simplification, we let $d = 10s$ (since $d = 10$ is not sufficient to setup the experiment, we let $d = 500$ when $s = 1$), and adopt the typical assumption that a route obtained by traceroute is a shortest path between the source and the destination [9]. Each sample obtained from one pair of (s, d) is denoted as $G^{T(s, d)}$. Table 1 shows the nine samples for Dimes-AS and UCLA-AS, respectively.

We first investigate the correlation between the original k-core index and the new k-core index (denoted as k_s^T) estimated from traceroute samples. As shown in Figure 1, for Dimes-AS with $T(528, 5280)$ and UCLA-AS with $T(764, 7640)$, most of the nodes have their k-core values estimated correctly; that is, they are distributed around the line $k_s = k_s^T$.

We also validate the robustness of k-core index by checking the attack sequence. For each sample $G^{T(s,d)}$ from $T(s,d)$, we obtain the list of nodes in the descending order of k-core values, denoted as $\zeta^{T(s,d)}$. For the nodes with a same k-core value, we reorder them by their degrees. Similarly, from the original network we can get the attack sequence ζ . We then measure the distance between the two sequences $\zeta^{T(s,d)}$ and ζ . We define the distance between two rank lists r_1 and r_2 with a same length as follows:

$$d_s = \frac{\sum_{\forall i,j(i \neq j)} d_{ij}}{n(n-1)}, \quad (3)$$

where n is the length of the rank list, and

$$d_{ij} = \begin{cases} 1 & r_1(i) > r_1(j), r_2(i) \leq r_2(j) \\ 1 & r_1(i) < r_1(j), r_2(i) \geq r_2(j) \\ 1 & r_1(i) = r_1(j), r_2(i) \neq r_2(j) \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

in which $r_1(i)$ ($r_2(i)$) stands for the rank of i in r_1 (r_2). Therefore, a lower d_s indicates the greater similarity between r_1 and r_2 . We select top f_{top} nodes from $\zeta^{T(d,s)}$ as r_1 , and select the same nodes from ζ to compose r_2 . As shown in Figure 2, for both the DIMES-AS and UCLA-AS networks, as the number of sources increases, d_s decreases rapidly. For example, in DIMES-AS, the sequence from $T(528, 5280)$ is very similar to the real sequence with $d_s < 0.1$. For UCLA-AS, the sample $T(764, 7640)$ also captures most of the real sequence information.

In summary, for the real-world AS-level Internet, the task of estimating the k-core value of an AS, or obtaining the attack sequence based on k-core index, is not that difficult. For DIMES-AS, the attackers only need to collect IP addresses from 528 (i.e., 2%) ASes to perform traceroute, and therefore is feasible in reality. For UCLA-AS, they may need to collect more IP addresses, say from 764 (i.e., 2%) ASes — but still feasible.

5 Empirical Study

In this section, we perform malicious attacks on real-world AS-level networks, and compare the results using the above-mentioned six measures. Some explanations will then be given to highlight the characteristic of a k-core attack.

5.1 Emperimental Results

Here, we consider the degree-preferred attack and k-core-preferred attack. We denote the fraction of removed nodes as f_r . For the four structural measures, we first perform one round of attack and then calculate the measure values. For the two propagation measures, we first conduct one wave of attack and then simulate the SIS or SIR model on the networks for 100 times, and return the average f_c^{SIS} or f_{max}^{SIR} value. Note that we let $\mu = 1.0$ and $\beta = 0.3$ for the SIS model, and $\mu = 1.0$ and $\lambda = 0.3$ for the SIR model.

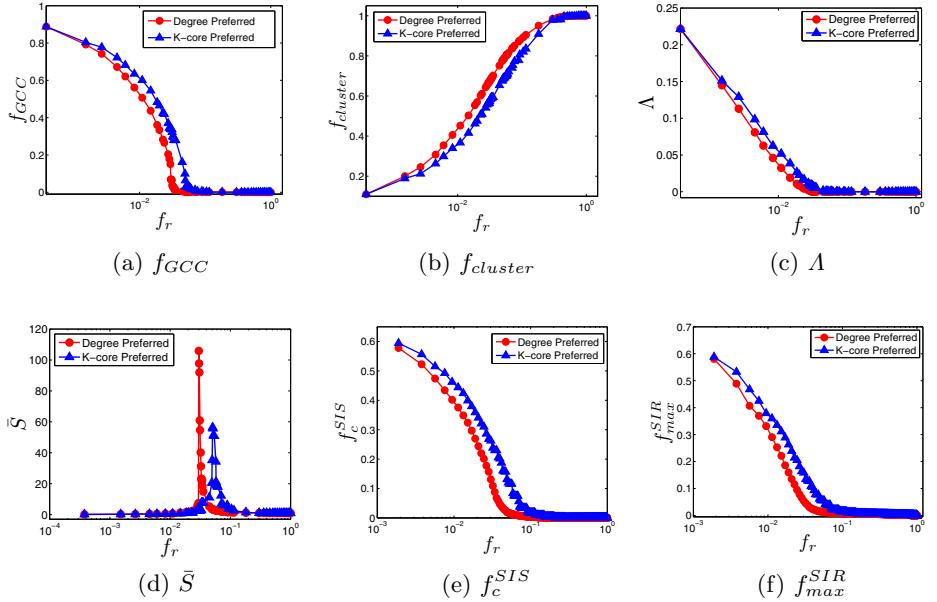


Fig. 3. Comparison of two attacks to DIMEAS-AS

Figures 3 and 4 show the results. As can be seen, to our surprise, we find that the k-core-preferred attack (CA) is less malicious to the AS-level Internet than the degree-preferred attack (DA). We take the DIMEAS-AS network for illustration. As shown in Figure 3a, as f_r increases, f_{GCC} decreases more slowly for CA. This means that after removing the same amount of nodes, the network damaged by CA contains a larger GCC . Meanwhile, $f_{cluster}$ increases more quickly for DA, which implies that DA is more likely to break the network into pieces. As to Λ , it decreases less steeply for CA as f_r grows. That is to say, compared with DA, CA will not degrade the network efficiency rapidly. Finally, regarding to \bar{S} , the critical points of f_r at which a phase transition occurs are different for CA and DA. Specifically, the critical point for CA is 0.051, a value much larger than 0.029, the critical point for DA. This implies that DA can result in an earlier collapse of the network. Indeed, additional 528 ASes need to be attacked for CA to collapse DIMEAS-AS, and this number rises to 1146 in UCLA-AS.

The propagation measures also validate the less maliciousness of k-core-preferred attack. As can be seen in Figure 3e, for the model $SIS(1.0, 0.3)$, f_c^{SIS} decreases more slowly for CA, which means that the information or disease will spread wider in the network bearing CA rather than DA. A similar trend can be found for f_{max}^{SIR} in Figure 3f with the $SIR(1.0, 0.3)$ model. Note that we have tried different configurations of μ , β and λ for the SIS and SIR models, and always obtained results similar to the ones in Figure 3e and Figure 3f.

All the six measures indicate a same result for the UCLA-AS network in Figure 4; that is, the k-core-preferred attack is less malicious than the degree-preferred attack. Nevertheless, it is still noteworthy that the measure differences between

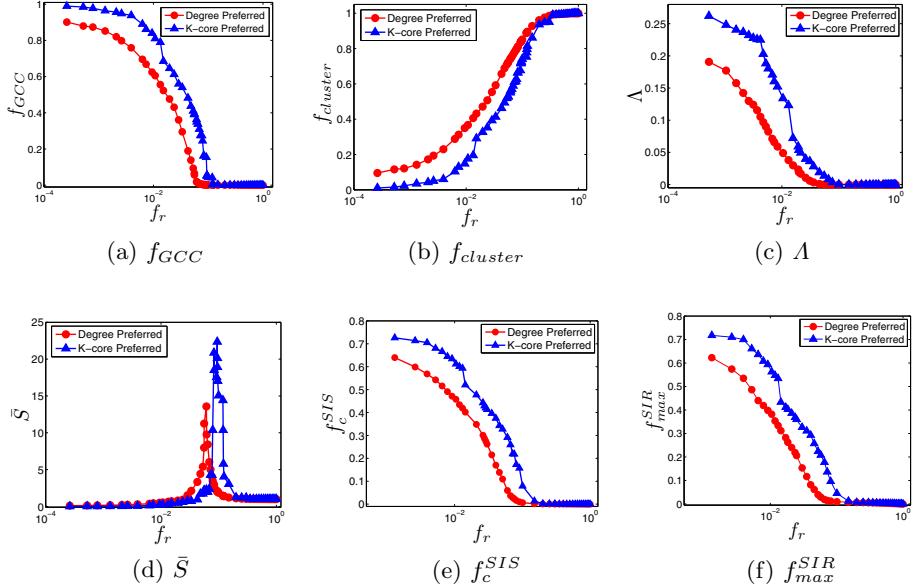


Fig. 4. Comparison of two attacks to UCLA-AS

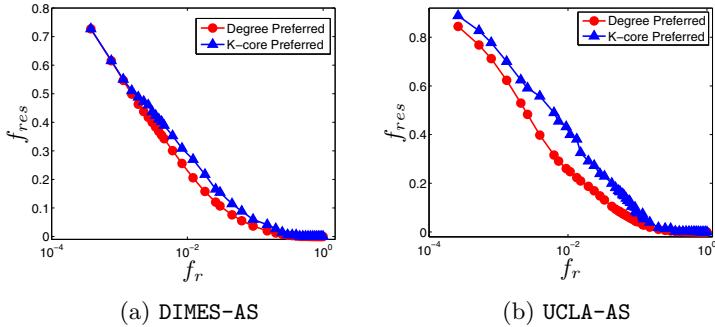


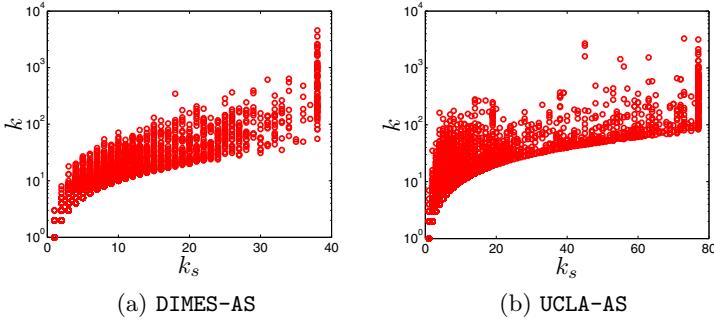
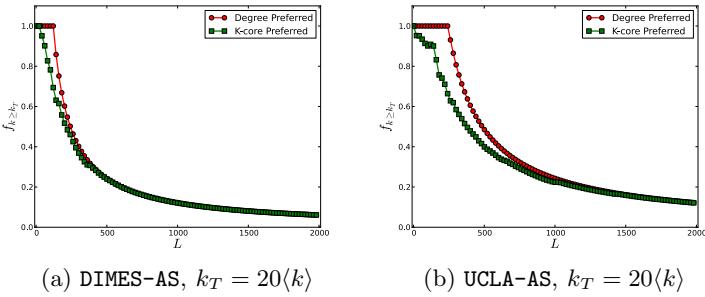
Fig. 5. The fraction of residual links

CA and DA are greater for the UCLA-AS network. For instance, as shown in Figure 4d, the critical points for CA and DA are 0.095 and 0.061 respectively, which lead to a gap larger than the one in the DIMES-AS network.

In summary, although being influential for information diffusion [14], the concept of k-core seems not that important for malicious attacks. In particular, the k-core-preferred attack is less malicious than the simple degree-preferred attack to the AS-level Internet.

5.2 Explanations and Discussions

Here, we try to explain the above finding by exploring the relationship between the degree and k-core of a node.

**Fig. 6.** Comparison of degrees in the same shell**Fig. 7.** Comparison of $f_{k \geq k_T}$

The essence of attacks based on node removals is to delete the links connected to those nodes. We define the fraction of residual links in the network as f_{res} and observe how it varies as f_r increases. As shown in Figure 5, the fraction of residual edges for the k-core-preferred attack is clearly larger than the one for the degree-preferred attack. This implies that the k-core-preferred attack leads to much less link deletions.

Let us take a closer look at the nodes with a same k-core value. As shown in Figure 6, for nodes in the same shell, their degrees vary dramatically. As a result, compared with the degree-preferred attack, the k-core-preferred attack tends to delete less links from the nodes that have higher k-core values but lower degrees. To further illustrate this, we compare the attack sequences of CA and DA. We choose the first L nodes from the sequences and examine the fraction of nodes with degrees no less than a threshold k_T , denoted as $f_{k \geq k_T}$. As shown in Figure 7, compared with the degree-preferred attack, $f_{k \geq k_T}$ is obviously less for the k-core-preferred attack, especially at the early stage when $50 < L < 300$ for DIMES-AS or $50 < L < 1000$ for UCLA-AS. It should also be noted that the gap of $f_{k \geq k_T}$ is greater in UCLA-AS, which could also explain the more evident differences between the two attacking strategies in Figure 3 and Figure 4.

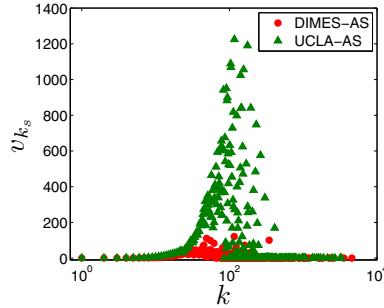


Fig. 8. Variance of k-core for nodes with a same degree

Moreover, to understand why the gap between the two different attacks is more obvious in UCLA-AS, we examine the variance of k-core for nodes with a same degree. Figure 8 shows the result. As can be seen, the variance in UCLA-AS is much higher than the variance in DIMEs-AS, especially when $50 < k < 200$. This implies that the attack sequences for DA and CA are more inconsistent in UCLA-AS, which eventually leads to significantly different attack effects.

In summary, the reason for the k-core-preferred attack being less malicious is that the nodes with high k-core values may own low degrees, and thus lead to less link deletions in the early stage of the attack.

6 Conclusion

The Internet plays a vital role in modern communications. However, as a typical instance of scale-free networks, it is fragile to the malicious attacks. In this paper, we proposed *k*-core-preferred attack, a malicious attack for nodes with higher k-core values, and compared it with the classic degree-preferred attack. Extensive experiments on two AS-level Internet topologies using six measures demonstrate that: (1) The k-core-preferred attack is feasible in real-world scenarios; (2) The k-core-preferred attack is less malicious than the degree-preferred attack; (3) The nodes in a same shell may have drastically different degrees, which degrades the efficiency of a k-core-preferred attack.

Acknowledgements. This work was partially supported by the fund of the State Key Laboratory of Software Development Environment under Grant SKLSDE-2011ZX-02, the Research Fund for the Doctoral Program of Higher Education of China under Grant 20111102110019, and the National 863 Program under Grant 2012AA011005. Jichang Zhao thanks the China Scholarship Council (CSC) for its support. Junjie Wu was supported in part by National Natural Science Foundation of China under Grants 71171007 and 70901002, by the Foundation for the Author of National Excellent Doctoral Dissertation of PR China under Grant 201189, and by the Program for New Century Excellent Talents in University under Grant NCET-11-0778.

References

1. Albert, R., Jeong, H., Barabási, A.L.: Error and attack tolerance of complex networks. *Nature* 406(6794), 378–382 (2000)
2. Boguñá, M., Papadopoulos, F., Krioukov, D.: Sustaining the internet with hyperbolic mapping. *Nature Communications* 1(62) (2010)
3. Brandes, U.: A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology* 25, 163–177 (2001)
4. Butler, K., Farley, T., McDaniel, P., Rexford, J.: A survey of bgp security issues and solutions. *Proceedings of the IEEE* 98, 100–122 (2010)
5. Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., Shir, E.: A model of internet topology using k-shell decomposition. *PNAS* 104(27), 11150–11154 (2007)
6. Cohen, R., Erez, K., Ben-Avraham, D., Havlin, S.: Resilience of the internet to random breakdowns. *Phys. Rev. Lett.* 85(21), 4626–4628 (2000)
7. Cohen, R., Erez, K., Ben-Avraham, D., Havlin, S.: Breakdown of the internet under intentional attack. *Phys. Rev. Lett.* 86(16), 3682–3685 (2001)
8. Cowie, J., Ogielski, A.T., Premore, B.J., Yuan, Y.: Internet worms and global routing instabilities. In: Proc. SPIE, vol. 4868 (2002)
9. Donnet, B., Friedman, T.: Internet topology discovery: a survey. *IEEE Communications Surveys and Tutorials* 9(4), 2–15 (2007)
10. Dorogovtsev, S.N., Goltsev, A.V., Mendes, J.F.F.: *k*-core organization of complex networks. *Phys. Rev. Lett.* 96, 040601 (2006)
11. Guillaume, J.L., Latapy, M., Magoni, D.: Relevance of massively distributed explorations of the internet topology: Qualitative results. *Computer Networks* 50, 3197–3224 (2006)
12. Holme, P., Kim, B.J., Yoon, C.N., Han, S.K.: Attack vulnerability of complex networks. *Phys. Rev. E* 65(5), 056109 (2002)
13. Huffaker, B., Plummer, D., Moore, D., Claffy, K.C.: Topology discovery by active probing. In: SAINT-W 2002, pp. 90–96 (2002)
14. Kitsak, M., Gallos, L.K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H.E., Makse, H.A.: Identification of influential spreaders in complex networks. *Nature Physics* 6, 888–893 (2010)
15. Kumpula, J.M., Onnela, J.P., Saramäki, J., Kaski, K., Kertész, J.: Emergence of communities in weighted networks. *Phys. Rev. Lett.* 99(22), 228701 (2007)
16. Latora, V., Marchiori, M.: Efficient behavior of small-world networks. *Phys. Rev. Lett.* 87(19), 198701 (2001)
17. Liljenstam, M., Yuan, Y., Premore, B.J., Nicol, D.M.: A mixed abstraction level simulation model of large-scale internet worm infestations. In: MASCOTS 2002, pp. 109–116 (2002)
18. Schneider, C.M., Moreira, A.A., Andrade Jr., J.S., Havlin, S., Herrmann, H.J.: Mitigation of malicious attacks on networks. *PNAS* 108(10), 3838–3841 (2011)
19. Seidman, S.B.: Network structure and minum degree. *Social Networks* 5, 269–287 (1983)
20. Shakkottai, S., Fomenkov, M., Koga, R., Krioukov, D., Claffy, K.: Evolution of the internet as-level ecosystem. *European Physical Journal B* 74, 271–278 (2006)
21. Zhang, G.Q., Zhang, G.Q., Yang, Q.F., Cheng, S.Q., Zhou, T.: Evolution of the internet and its cores. *New J. Phys.* 10(12), 123027 (2008)
22. Zhang, J., Zhao, H., Xu, J., Liu, Z.: Characterizing and modeling the internet router-level topology - the hierarchical features and hir model. *Comput. Commun.* 33, 2001–2011 (2010)

Fuzzy Keyword Search over Encrypted Data in the Public Key Setting

Qiuxiang Dong^{1,2,3}, Zhi Guan^{1,2,3,*}, Liang Wu⁴, and Zhong Chen^{1,2,3}

¹ Institute of Software, School of EECS, Peking University, China

² MoE Key Lab of High Confidence Software Technologies (PKU)

³ MoE Key Lab of Network and Software Security Assurance (PKU)

⁴ Computer Network Information Center, Chinese Academy of Sciences

{dongqx, guanzhi, chen}@infosec.pku.edu.cn, wuliang@cnic.cn

Abstract. Searchable encryption is used to support searches over encrypted data stored on cloud servers. Traditional searchable encryption only supports exact keyword search instead of more flexible fuzzy keyword search. To solve this problem, a recent emerging paradigm, named fuzzy keyword searchable encryption, has been proposed. There have been some proposals designed for fuzzy keyword search in the symmetric key setting, but none efficient schemes in the public key setting. In this paper, we propose a new primitive of interactive public key encryption with fuzzy keyword search (IPEFKS), which supports efficient fuzzy keyword search over encrypted data in the public key setting. We construct and implement a homomorphic encryption based IPEFKS scheme. To compare this scheme with the existing ones, we implement LWW-FKS, which, to the best of our knowledge, is the most efficient among the existing schemes. The experimental results show that IPEFKS is much more efficient than LWW-FKS.

Keywords: Fuzzy Keyword Search, Public key Encryption with Keywords Search, Cloud Computing.

1 Introduction

1.1 Background

In recent years, due to the appealing features of cloud computing, more and more data have been centralized into cloud. To protect clients' privacy, data with sensitive information are usually stored in the encrypted form. However, encrypted storage makes it hard to perform searches over the data. To cope with this problem, various techniques for searching over encrypted data, namely searchable encryption, have been proposed in the literature [1–4].

Searchable encryption schemes can be divided into two categories [14], the symmetric key searchable encryption (SSE) [3, 4] and the public key searchable encryption (PSE) [1, 2]. In SSE, the data sender and data receiver are the same

* Corresponding author.

entity or different entities sharing the same secret key. While in PSE, they are different entities and may not share a secret key. By comparison, SSE is more efficient than PSE, while PSE can be deployed in scenarios where it is infeasible to share a common secret key between different entities.

One of the drawbacks of traditional searchable encryption schemes, both SSE and PSE, is that they only support exact keyword search. This will affect system usability, since typos and format inconsistencies often occur when users input keywords. To enhance system usability, searchable encryption schemes with fuzzy keyword search capability [9] are necessary.

In the symmetric key setting, some efficient proposals have been designed for fuzzy keyword search over encrypted data [12, 13, 15, 16]. However, in the public key setting, to the best of our knowledge, there have been only two researches on achieving this functionality [5, 9]. In addition, we find that both schemes are not efficient for real applications. In this work, we construct and implement a homomorphic encryption based IPEFKS scheme, which achieves much higher efficiency compared with these two schemes.

1.2 Contributions

Our contributions in this paper are threefold.

1. We propose the primitive of IPEFKS that supports fuzzy keyword search over encrypted data and define its security requirement. We give a construction based on homomorphic encryption and prove that it is secure under the assumption that there exists an IND-CPA secure homomorphic encryption scheme.
2. We leverage a nice property of bilinear pairings in such a way that the cloud server can build an inverted index over encrypted data without knowing the encrypted keywords. Since an inverted index supports much more efficient search than a forward index when the number of documents is large, we may find uses for this approach in other applications.
3. We evaluate the efficiency of the homomorphic encryption based IPEFKS scheme by implementing the FV.SH homomorphic encryption scheme [8]. To show the efficiency advantage of IPEFKS over the other existing schemes, we implement the scheme proposed by Li *et al.* [9], which, to the best of our knowledge is the most efficient scheme among the existing ones. We name this scheme LWW-FKS for simplicity. The experimental results demonstrate that IPEFKS is much more efficient than it.

1.3 Organization

The rest of this paper is organized as follows. In section 2, we present the related work on fuzzy keyword searchable encryption. In section 3, we give the definition of IPEFKS and a concrete construction of it. In section 4, we show how to allow the cloud server to build an inverted index over encrypted data in a secure way. In section 5, we compare the efficiency of IPEFKS with LWW-FKS.

In section 6 we show security advantage of IPEFKS over LWW-FKS. Section 7 concludes this paper.

2 Related Work

Our work is built on fuzzy keyword searchable encryption. The most important related researches in this field are discussed below.

Fuzzy keyword search (FKS) over encrypted data in the public key setting is firstly proposed by Li *et al.* in [9]. Given a keyword q , FKS intends to find the encrypted documents containing keywords that are within a certain edit distance from q . The edit distance between two words w_1 and w_2 is the number of operations, including substitution, deletion and insertion, required to transform one of them into the other. Given a keyword, the basic idea of [9] is to enumerate all wildcard-based keywords that are within a predefined edit distance to it. Therefore, this scheme transforms a single fuzzy keyword search operation into several exact keyword search operations. The succeeding work done by Wang *et al.* [15] focuses on efficiency. They present a symmetric fuzzy keyword searchable encryption scheme with trie-traverse searching index in order to achieve high efficiency.

As discussed in [13], both schemes cited above can not use any other wildcard patterns than the ones prepared by the sender because the wildcard is realized by exact keyword search. The authors of [13] split a keyword into several characters and can achieve more splendid keyword matching patterns, e.g., wildcard search, partial matching. However, as for FKS, they give the same wildcard-based approach as [9]. Another recent work on symmetric FKS has been presented in [12]. The authors construct a symmetric similarity keyword search scheme using a fuzzy extractor and a protocol for computing Hamming distance. Actually, all above mentioned schemes do not support fuzzy keyword search in the real sense. The key issue to design FKS schemes is computing the edit distance between two encrypted keywords. To solve this problem, the authors of [16] and [5] share the common idea of translating metric space. Concretely, they translate edit distance into a new metric space with locality sensitive function family.

Except for LWW-FKS, the scheme proposed in [5] is the other FKS scheme in the public key setting among all others mentioned above. It is based on the embedding algorithm in [10] and the similarity search scheme in [6]. It uses PIS (Private Information Storage) [11] and PIR (Private Information Retrieval) [7], both of which are interactive cryptographic protocol with low efficiency, especially when used to store and retrieve documents consisting of a large number of bits. This makes this scheme inefficient for encrypted document retrieval.

Note that another research [18] sharing a similar name with ours solves the problem of keyword guessing attack but not the problem of fuzzy keyword search, which is the focus of this paper. In conclusion, the only somewhat practical FKS scheme in the public key setting is LWW-FKS. In this paper, we present a much more efficient FKS scheme than it.

3 IPEFKS

3.1 Preliminary

We denote vectors by lower-case bold italic letters, say \mathbf{x} , \mathbf{y} , \mathbf{z} , etc. We assume all keywords are strings of the same length, if this is not the case, we append some wildcard symbols (e.g., $*$) to the shorter ones.

The authors of [10] show that there exist $0 < \alpha < \beta < c_2$ and an embedding Ψ from $\{0, 1\}^N$ with edit distance ed to $\{0, 1\}^{c_2(\log_2(1/\delta))}$ with Hamming distance \mathcal{HD} such that:

- If $ed(\mathbf{x}, \mathbf{y}) \leq t$, then $\mathcal{HD}(\Psi(\mathbf{x}), \Psi(\mathbf{y})) \leq \alpha \log_2(1/\delta)$.
- If $ed(\mathbf{x}, \mathbf{y}) \geq 2^{c_1(\sqrt{\log_2 N \log_2 \log_2 N})} t$, then $\mathcal{HD}(\Psi(\mathbf{x}), \Psi(\mathbf{y})) \geq \beta \log_2(1/\delta)$.

The above fact indicates that we can use Hamming distance to represent edit distance. Therefore, we focus on searchable encryption with Hamming distance as the distance measurement.

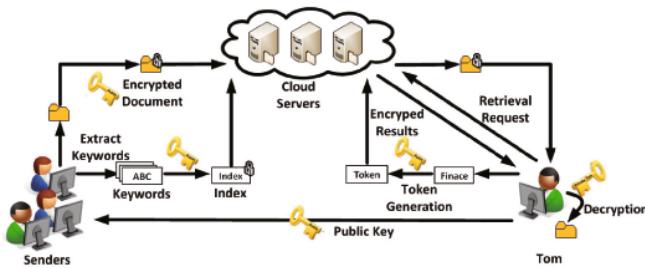


Fig. 1. Work flow of the IPEFKS scheme in the public key setting

3.2 Definition of IPEFKS

Definition 1. The IPEFKS scheme consists of the following five polynomial time algorithms, i.e., **KeyGen**, **IPEFKS**, **TokenGen**, **Search** and **Retrieve**, defined as follows:

- **KeyGen**(1^λ): Takes a security parameter λ as input, and generates a public/private key pair pk, sk .
- **IPEFKS**(pk, w): Takes a public key pk and a keyword w as inputs, and produces a searchable encryption of w .
- **TokenGen**(pk, v): Given a public key pk and a query keyword v , this algorithm produces a token T_v .
- **Search**(T_v, S_w): With a token $T_v = \text{TokenGen}(pk, v)$ and a keyword ciphertext $S_w = \text{IPEFKS}(pk, w)$, outputs an encrypted result ER_w .
- **Retrieve**(sk, ER_w, th): Given a private key sk , the encrypted result ER_w of the keyword w , and a Hamming distance threshold th , outputs YES if $\mathcal{HD}(v, w) < th$, otherwise, outputs NO.

The work flow of the IPEFKS scheme is illustrated in **Figure.1**. The receiver Tom runs the **KeyGen** algorithm to generate his public/private key pair pk, sk and announces pk publicly. When a sender wants to send a document to Tom. She/he extracts the keywords of the document. Then the sender runs the **IPEFKS** algorithm to encrypt every keyword under Tom's public key, and these encrypted keywords form the encrypted index of the document. The index and the encrypted documents are then sent to the cloud server. When Tom wants to retrieve documents containing keywords within a certain Hamming distance threshold compared with his interested keyword "Finace" (i.e., Tom omits an 'n' of the keyword "Finance"), he runs the **TokenGen** algorithm to generate a *token* and sends it to the cloud server. Upon receiving this query request, the cloud server runs the **Search** algorithm and sends the encrypted results back to Tom, who then runs the **Retrieve** algorithm and returns the retrieval request back to the server. Finally, the encrypted documents are returned back to Tom. Tom uses his private key to decrypt the encrypted documents. Note that we do not specify how the documents are encrypted because it is unrelated with the search functionality.

We treat the cloud server as an honest-but-curious attacker in the security model and give a rigorous security definition as follows:

Definition 2. An IPEFKS scheme is semantically secure against an adaptive chosen keyword attack if every PPT (Probabilistic Polynomial Time) attacker has a negligible advantage in the following attack game.

1. **Setup.** The challenger runs the **KeyGen** algorithm and obtains a key pair (pk, sk) and sends pk to the attacker.
2. **Phase 1.** The attacker can ask for arbitrarily many keyword ciphertexts and query tokens from the challenger (the attacker does not know the associated keywords). In addition, the attacker can choose a query token and ask the challenger for the search results of the chosen query.
3. **Challenge.** At some point, the attacker sends the challenger two equal-length keywords $\mathbf{w}_0, \mathbf{w}_1$, on which it wants to be challenged. The challenger picks a random bit $b \in \{0, 1\}$ and gives the attacker the keyword ciphertext $C = \text{IPEFKS}(pk, \mathbf{w}_b)$ and the token $T = \text{TokenGen}(pk, \mathbf{w}_b)$.
4. **Phase 2.** The attacker can continue to ask for more keyword ciphertexts, query tokens and search results as in **Phase 1**.
5. **Guess.** The attacker outputs a guess bit b' for b . The advantage of an adversary \mathcal{A} is defined to be $\text{Adv}_{\mathcal{A}} = |\Pr[b = b'] - 1/2|$.

3.3 Construction

In this section, we give a construction of the IPEFKS scheme based on a homomorphic encryption scheme, named FV.SH encryption scheme [8]. Before presenting the concrete construction, we give some intuitions. The Hamming distance of two binary strings $\mathbf{x} = \mathbf{x}_1 \cdots \mathbf{x}_m$ and $\mathbf{y} = \mathbf{y}_1 \cdots \mathbf{y}_m$ is $\sum_{i=1}^m (\mathbf{x}_i \oplus \mathbf{y}_i)$.

Our construction leverages a nice property of the FV.SH encryption scheme, that is when the plaintexts are polynomials with coefficients in Z_2 , the addition operation is equal to the XOR operation.

The FV.SH encryption scheme used in the following construction consists of four polynomial time algorithms, the key generation algorithm **KeyGen**, the encryption algorithm **Enc**, the decryption algorithm **Dec** and the ciphertext addition algorithm **Add**. Detailed description of these algorithms are given in section 5.1. We don't care about the ciphertext Multiplication algorithm, since it is unrelated with our work. The detailed construction is shown below.

- **KeyGen**(1^λ): Given a security parameter λ , generates a public/private key pair, pk and sk , of the FV.SH encryption scheme.
- **IPEFKS**(pk, w): Given an m -bit keyword $w = w_1 w_2 \dots w_m$, encodes the keyword w as a polynomial w (we reuse the notation for simplicity) in $Z_2[x]/x^d + 1$, where d is a parameter in the FV.SH encryption scheme, and computes $S_w = \text{Enc}_{pk}(w)$.
- **TokenGen**(pk, v): Given an m -bit keyword $v = v_1 v_2 \dots v_m$, encodes the keyword v as a polynomial v in $Z_2[x]/x^d + 1$ and computes $T_v = \text{Enc}_{pk}(v)$.
- **Search**(T_v, S_w): Computes the encrypted result ER_w by adding T_v and S_w by running the **Add** algorithm of the FV.SH encryption scheme.
- **Retrieve**(sk, ER_w, th): Decrypts ER_w by running $\text{Dec}_{sk}(ER_w)$, adds the coefficients of $\text{Dec}_{sk}(ER_w)$, and sets the result to $\mathcal{HD}(w, v)$. If $\mathcal{HD}(w, v) < th$, outputs YES, otherwise outputs NO.

In the **IPEFKS** and **TokenGen** algorithm, we encode the keyword by taking each bit of the keyword as the coefficient of an $(m - 1)$ -degree polynomial.

3.4 Correctness and Security

Lemma 1. Given two binary vectors $w = w_1 \dots w_m$ and $v = v_1 \dots v_m$, the Hamming distance of w and v equals $\sum_{i=1}^m w_i \oplus v_i$, where the notation \oplus denotes the XOR operation.

Correctness is guaranteed by the property of the FV.SH encryption scheme, i.e., when the keywords are encoded as polynomials in $Z_2[x]/x^d + 1$, the addition operation is equal to the XOR operation.

Theorem 1. The IPEFKS scheme is semantically secure against a chosen keyword attack (CKA) if the FV.SH homomorphic encryption scheme is IND-CPA secure.

Proof: Suppose there exists an adversary \mathcal{A} that has a non-negligible advantage ϵ in breaking the IPEFKS. We show that there is an adversary \mathcal{B} that can achieve the same advantage in winning a chosen plaintext attack (CPA) game for attacking the IND-CPA secure FV.SH encryption scheme. Therefore, we have a contradiction. We can conclude that \mathcal{A} cannot exist and thus that IPEFKS is semantically secure against a chosen keyword attack (CKA).

In the chosen plaintext attack, where \mathcal{C} is the challenger, the adversary \mathcal{B} is supposed to give two messages m_0, m_1 to \mathcal{C} . It then receives an encryption $C_{IND-CPA} = \text{Enc}_{pk}(m_b)$ from \mathcal{C} , where b is a random bit chosen by \mathcal{C} . \mathcal{B} outputs a guess bit b' and wins if $b' = b$.

In the chosen keyword attack, \mathcal{B} works as a simulator who acts as the challenger. The adversary \mathcal{A} outputs two messages m_0 and m_1 ($m_0 \neq m_1$). \mathcal{B} sends m_0 and m_1 to \mathcal{C} , who then returns a challenge $C_{IND-CPA} = \text{Enc}_{pk}(m_b)$; \mathcal{B} passes this challenge to \mathcal{A} . Finally \mathcal{A} gives a guess b' for b , \mathcal{B} outputs b' as its guess bit for the CPA game described above.

As for queries from \mathcal{A} in Phase 1 and Phase 2, \mathcal{B} maintains two lists, L_1, L_2 , which are initially empty. The two lists consist of tuples $\langle \mathbf{w}_i, \text{Enc}_{pk}(\mathbf{w}_i) \rangle$. When \mathcal{A} asks for a keyword ciphertext, \mathcal{B} responds by sending $\text{Enc}_{pk}(\mathbf{w})$ for a randomly chosen keyword \mathbf{w} and appends the tuple $\langle \mathbf{w}, \text{Enc}_{pk}(\mathbf{w}) \rangle$ to the list L_1 . When \mathcal{A} asks for a token, \mathcal{B} responds by sending $\text{Enc}_{pk}(\mathbf{w})$ for a randomly chosen keyword \mathbf{w} and appends the tuple $\langle \mathbf{w}, \text{Enc}_{pk}(\mathbf{w}) \rangle$ to the list L_2 . When \mathcal{A} sends back a token and asks for the search results, \mathcal{B} searches for the token $\text{Enc}_{pk}(\mathbf{w})$ and the corresponding keyword \mathbf{w} in the list L_2 and then computes the Hamming distance between \mathbf{w} and the keywords in the list L_1 and returns back $\text{IPEFKS}(pk, \mathbf{w}_i)$ satisfying $\mathcal{HD}(\mathbf{w}, \mathbf{w}_i) \leq th$, where th is chosen by \mathcal{B} . If the token sent by \mathcal{A} is not in the list L_2 , then \mathcal{B} returns \perp , and the CKA game exits.

The attacker \mathcal{A} cannot distinguish whether it is interacting with a real searcher or the simulator \mathcal{B} since the message distribution are the same. The advantage \mathcal{B} gains in the CPA game is the same as that of \mathcal{A} in the CKA game. Since FV.SH is IND-CPA secure, which indicates that the advantage gained by \mathcal{B} must be negligible, we get a secure IPEFKS scheme.

4 Building an Inverted Index over Encrypted Data

In the construction described in section 3.3, we do not specify the index structure of the encrypted documents. When there are a large number of documents, inverted index structure is preferable to forward index structure [19]. In this section, we show how to enable the cloud server to build an inverted index over encrypted data in a secure way.

4.1 Preliminary

Bilinear pairings: Let \mathbf{G}_1 and \mathbf{G}_2 be two cyclic multiplicative group of prime order p , g be a generator of the group \mathbf{G}_1 and $e : \mathbf{G}_1 \times \mathbf{G}_1 \longrightarrow \mathbf{G}_2$ be a bilinear map between them. The map satisfies the following properties:

- 1.Computable: given $u, v \in \mathbf{G}_1$ there is a polynomial time algorithm to compute $e(u, v) \in \mathbf{G}_2$.
- 2.Bilinear: for any integers $x, y \in [1, p]$ we have $e(g^x, g^y) = e(g, g)^{xy}$.
- 3.Non-degenerate: $e(g, g)$ is a generator of \mathbf{G}_2 .

4.2 Construction

The cloud server publishes the bilinear pairing parameters $(p, \mathbf{G}_1, \mathbf{G}_2, e, g)$. The cloud server and senders process as follows:

Sender: To send a document with identifier ID and keywords set $u(ID) = \{w_1, w_2, \dots, w_n\}$, the sender processes as follows:

1. Gets the public parameters, including the public key pk of the receiver, the bilinear pairing parameters $(p, \mathbf{G}_1, \mathbf{G}_2, e, g)$ of the cloud server;
2. Generates a random number $r \in Z_p^*$ and evaluates $(CA_i, CB_i) = (g^r, w_i^r)$, where CA_i and CB_i is the corresponding term for the i^{th} keyword in the ciphertext, which will be used by the cloud server to build an inverted index without disclosing the keywords privacy.
3. Runs the keyword encryption algorithm **IPEFKS** to encrypt every keyword in the set $u(ID)$;
4. Sends $CA_i = g^r$, $CB_i = w_i^r$, ID, and the searchable encrypted keywords **IPEFKS**(pk, w_i) (for $i = 1, 2, \dots, n$) together with the encrypted document to the cloud server.

Cloud Server: Assume that there exist m terms in the inverted index. The j^{th} term is $(SA_j = g^{r_j}, SB_j = w_j^{r_j}, \mathbf{IPEFKS}(pk, w_j), D_j)$, where D_j is the set of document identifiers such that every $d \in D_j$ contains the keyword w_j . Upon receiving a sending request from the sender, the cloud server processes as follows:

1. Decides whether w_i is equal to one of the m keywords by checking whether the equation $e(CA_i, SB_j) = e(SA_j, CB_i)$ is right for a certain j , where $j \in \{1, \dots, m\}$;
2. Adds the document identifier ID to the set D_j if the equation $e(CA_i, SB_j) = e(SA_j, CB_i)$ holds for a certain j , otherwise establishes a new quadruple, i.e., $(SA_{m+1} = CA_i, SB_{m+1} = CB_i, \mathbf{IPEFKS}(pk, w_i), D_{m+1})$ for the new keyword w_i , where $D_{m+1} = \{ID\}$.

Correctness of this construction is based on the bilinear property of bilinear pairings and security relies on the fact that Diffie-Hellman problem on the \mathbf{G}_2 group is hard. Because of the constrained space, we do not give a rigorous security proof here.

5 Performance Analysis

In this section, we evaluate the efficiency of the IPEFKS scheme by implementing the recently proposed FV.SH encryption scheme and comparing with LWW-FKS. We use the notations illustrated in Table 1 in the remaining part of this paper.

Table 1. Notation Table

Notation	Description	Notation	Description
d	$d = 2^k$ for some k , and is the largest bit-length of the embedded keywords output by the embedding algorithm Ψ	q	the fixed modulo
$\chi = D_{\mathbb{Z}^d, \sigma}$	denotes the discrete Gaussian distribution with standard deviation σ over \mathbb{R}	t	the fixed modulo
R	$R = \mathbb{Z}/f(x)$, \mathbb{Z} is the set of integers and $f(x) = x^d + 1$, R is the set of polynomials whose degree is not larger than d	$\lfloor x \rfloor$	rounding down
R_q	the set of polynomials in R with coefficients in $(-q/2, q/2]$	$\lfloor x \rfloor$	rounding to the nearest integer
R_t	the set of polynomials in R with coefficients in $(-t/2, t/2]$	Δ	$\Delta = \lfloor q/t \rfloor$

5.1 Implementation of the FV.SH Encryption Scheme

In the FV.SH encryption scheme, ciphertexts consist of polynomials in the ring R_q . Plaintexts are polynomials in the ring R_t . d, q, t and σ are system parameters chosen in such a way that correctness and security are guaranteed. The FV.SH encryption scheme consists of the following algorithms:

- **FV.SH.KeyGen(1^λ)**: Samples secret key $sk \xleftarrow{R} R_2$ uniformly. Samples $p_1 \xleftarrow{R} R_q$ uniformly and an error $e \xleftarrow{R} \chi$. Computes the public key $pk = (p_0 = [-(p_1 s + e)]_q, p_1)$
- **FV.SH.Enc(pk, m)**: Samples $u \xleftarrow{R} R_2$, $e_1, e_2 \xleftarrow{R} \chi$ uniformly and returns $ct = (ct[0] = [p_0 u + e_1 + \Delta \cdot m]_q, ct[1] = [p_1 u + e_2]_q)$
- **FV.SH.Add(ct_1, ct_2)**: Returns $([ct_1[0] + ct_2[0]]_q, [ct_1[1] + ct_2[1]]_q)$
- **FV.SH.Dec(ct)**: Computes $\left\lfloor \frac{[ct[0] + sk \cdot ct[1]]_q}{\Delta} \right\rfloor_t$

We set two tuples of parameters for the FV.SH encryption scheme. Both tuples provide 128 bits of security with distinguishing advantage 2^{-64} . In addition, some optimizations (e.g., using the bounded discrete Gaussian distribution to replace the real discrete Gaussian distribution) are taken. For interested readers, please refer to [8] for details. We implement the scheme in C using FLINT, namely Fast Library for Number Theory [21]. We test the code on an Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz running Linux 3.2.0-34-generic x86_64. The efficiency of the FV.SH encryption scheme is shown in Table 2.

5.2 Efficiency Comparison

As we have shown in section 2, the proposal shown in [5] is not practical for encrypted document retrieval. As a result, in this section, we only need to compare IPEFKS with LWW-FKS. To cope with the scenario where there are a large number of documents, we use an inverted index here. The process of building an

Table 2. Timing in *ms* for FV.SH operations key generation, encryption, decryption and homomorphic addition in C. Parameters (P_1) and (P_2) have 128 bits of security with distinguishing advantage 2^{-64} .

Parameters	FV.SH.KeyGen	FV.SH.Enc	FV.SH.Dec	FV.SH.Add
(P_1) $d = 4096$ $\sigma = 16, q = 2^{128}$	80	33	7	3
(P_2) $d = 8192$ $\sigma = 8, q = 2^{128}$	153	60	15	6

inverted index over encrypted data is shown in section 4. Without loss of generality, we assume the average keyword length to be 10 and the number of typos in the searcher’s inputs is smaller than 4, which we think is reasonable when the keyword length is 10. The exact keyword searchable encryption scheme used in LWW-FKS is PEKS [1], which takes the type A parameter in the PBC library [20]. In LWW-FKS, the computation cost of the search algorithm, keyword encryption algorithm and token/trapdoor generation algorithm is proportional to $O(10^t kn)$, while the computation cost in IPEFKS is only linear in kn , where kn denotes the keyword number and t denotes the edit distance threshold.

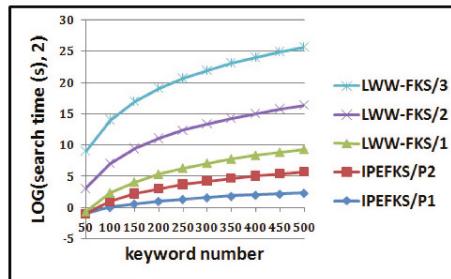


Fig. 2. Search time comparison between IPEFKS and LWW-FKS

Figure.2 shows the experimental results of the search time spent by the cloud server in processing a keyword search request. It can be seen that when the edit distance gets larger, the search time of LWW-FKS increases greatly, while the search time of IPEFKS is only related with the keyword number but not the edit distance.

Figure.3 and Figure.4 present the keyword encryption time and token(for IPEFKS)/trapdoor(for LWW-FKS) generation time respectively. When the edit distance gets larger, in LWW-FKS, more keyword encryptions are performed and more trapdoors are generated, thus leading to increasing time consumption. While, in IPEFKS, the edit distance does not affect these two processes, therefore only constant time is needed. The above experimental results demonstrate that IPEFKS is much more efficient than LWW-FKS.

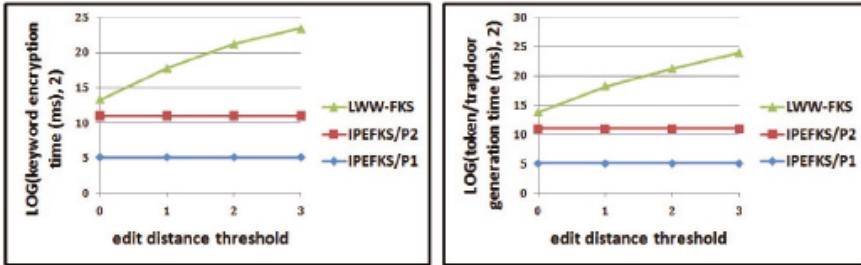


Fig. 3. Keyword encryption time and token/trapdoor generation time comparison

6 Security Analysis

IPEFKS is resistant against keyword guessing attack [17], which poses a great threat to clients' data privacy. While LWW-FKS suffers from this attack, because in LWW-FKS, cloud server has the capability of deciding whether a keyword is associated with the trapdoor received from the searcher independently. Specifically, given a token associated with a keyword W , the cloud server can encrypt a guessed keyword q and then runs the test algorithm of LWW-FKS to check whether $W = q$. When the cardinality of the keyword set is only polynomial in the security parameter, which is indeed the case in real applications, the cloud server can implement this attack successfully. In IPEFKS, the cloud server cannot test whether a keyword is associated with the trapdoor received from a searcher all by itself, because the **Retrieval** algorithm takes the secret key as an input, which is only known by the clients themselves.

7 Conclusion

In this paper, we present a new primitive, named IPEFKS, to solve the problem of fuzzy keyword search over encrypted data. By comparison, IPEFKS is not only more efficient but also more secure than all existing schemes. Moreover, to enable the cloud server to build an inverted index over encrypted data, we propose an approach, which leverages a nice property of bilinear pairings. To the best of our knowledge, we are the first to implement the FV.SH encryption scheme in C, which we think may be of independent interest for other works. Future work includes enhancing the efficiency of the FV.SH encryption scheme by using SIMD operations and other optimizations and giving experimental results on the real datasets by implementing the prototype of the system.

Acknowledgment. This work is partially supported by the HGJ National Significant Science and Technology Projects under Grant No. 2012ZX01039-004-009, Key Lab of Information Network Security, Ministry of Public Security under Grant No.C11606.

References

1. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
2. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and Efficiently Searchable Encryption. In: Menezes, A. (ed.) *CRYPTO 2007*. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007)
3. Goh, E.J.: Secure indexes. IACR Cryptology ePrint Archive 2003, 216 (2003)
4. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
5. Bringer, J., Chabanne, H.: Embedding edit distance to enable private keyword search. Human-centric Computing and Information Science 2(2) (2012)
6. Bringer, J., Chabanne, H., Kindarji, B.: Error-tolerant searchable encryption. In: Proceedings of the 2009 IEEE International Conference on Communications (2009)
7. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
8. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive 2012, 144 informal publication (2012)
9. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: INFOCOM 2010, pp. 441–445 (2010)
10. Ostrovsky, R., Rabani, Y.: Low distortion embeddings for edit distance. J. ACM 54(5) (October 2007)
11. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC 1997, pp. 294–303. ACM, New York (1997)
12. Pang, X., Yang, B., Huang, Q.: Privacy-preserving noisy keyword search in cloud computing. In: Chim, T.W., Yuen, T.H. (eds.) *ICICS 2012*. LNCS, vol. 7618, pp. 154–166. Springer, Heidelberg (2012)
13. Suga, T., Nishide, T., Sakurai, K.: Secure keyword search using bloom filter with specified character positions. In: Takagi, T., Wang, G., Qin, Z., Jiang, S., Yu, Y. (eds.) *ProvSec 2012*. LNCS, vol. 7496, pp. 235–252. Springer, Heidelberg (2012)
14. Tang, Q.: Search in encrypted data: Theoretical models and practical applications. IACR Cryptology ePrint Archive 2012, 648 (2012)
15. Wang, C., Ren, K., Yu, S., Urs, K.M.R.: Achieving usable and privacy-assured similarity search over outsourced cloud data. In: INFOCOM, pp. 451–459 (2012)
16. Kuzu, M., Islam, M.S., Kantarcioğlu, M.: Efficient similarity search over encrypted data. In: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, ICDE 2012. IEEE Computer Society, Washington, DC (2012)
17. Yau, W.-C., Heng, S.-H., Goi, B.-M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J. (eds.) *ATC 2008*. LNCS, vol. 5060, pp. 100–105. Springer, Heidelberg (2008)
18. Xu, P., Jin, H., Wu, Q., Wang, W.: Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. IEEE Transactions on Computers 99(PrePrints), 1 (2012)
19. Information Retrieval: Implementing and Evaluating Search Engines. MIT Press, Cambridge (2010) ISBN 978-0-262-02651-2
20. Lynn, B.: Pairing-Based Cryptography Library,
<http://crypto.stanford.edu/pbc/>
21. Hart, W.: FLINT: Fast Library for Number Theory, <http://www.flintlib.org>

The Hardness of (ϵ, m) -anonymity

Yujia Li¹, Dong Li², Xianmang He^{3,*}, Wei Wang¹, and Huahui Chen³

¹ School of Computer Science and Technology, Fudan University,
No.220, Handan Road, Shanghai, 200433
`{071021056,weiwang1}@fudan.edu.cn`

² Information Center, National Natural Science Foundation of China
No.83 Shuangqing Road, Haidian District, Beijing 100085
`lidong@nsfc.gov.cn`

³ School of Information Science and Engineering, NingBo University, 315122
`{chenhuahui,hexianmang}@nbu.edu.cn`

Abstract. When a table containing individual data is published, disclosure of sensitive information should be prohibitive. (ϵ, m) -anonymity was a new anonymization principle for preservation of proximity privacy, in publishing numerical sensitive data. It is shown to be NP-Hard to (ϵ, m) -anonymize a table minimizing the number of suppressed cells. Extensive performance study verified our findings that our algorithm is significantly better than the traditional algorithms presented in the paper[1].

Keywords: privacy preservation, algorithm, proximity privacy.

1 Introduction

With the growing use of public services, privacy preservation attracts increasing attention from users, industry, and the research community. A bulk of research in this area has been devoted to central publication. In that scenario, a publisher aims at releasing an anonymized version T^* of a microdata table T , so that researchers can use T^* to derive statistical results about T , whereas no malicious user, called an adversary, can infer the sensitive data of any individual from T^* . Consider, for instance, that a bank wants to publish its payment records in Table 1, called the microdata. Attribute Salary is sensitive, that is, the publication must ensure that no adversary can accurately infer the salary of any employee. “Age” and “Zipcode” are quasi-identifier (QI) attributes, because they can be utilized in combination to reveal the identity of an individual.

The proper measures must be taken to ensure that its publication does not endanger the privacy of the individuals that contributed the data. One way to overcome above threat is suppression. In a typical suppression-based solution, the microdata table was partitioned into QI-groups, and then converts the QI values in each group to the same form, e.g., replaces distinct values on each QI attribute with stars. Table 2 is such an example by suppressing Table 1 based on a partition of three QI-groups.

* Corresponding author.

Table 1. Microdata

Name	Age	Zip	Salary
Bob	20	12k	1020
Lily	21	12k	1030
Jane	22	12k	10000
Jame	24	16k	10000
Alex	24	24k	18000
Lucy	24	24k	24000
Andy	39	36k	31000
Sarah	45	36k	33000
John	39	36k	45000

Table 2. Suppression

GID	Age	Zip	Salary
1	*	12k	1020
1	*	12k	1030
1	*	12k	10000
2	24	*	10000
2	24	*	18000
2	24	*	24000
3	*	36k	31000
3	*	36k	33000
3	*	36k	45000

Assumes that each tuple in the group has an equal chance of being owned by Bob. Without further information, the adversary concludes that Bob's salary is in the short interval [1020, 1030] with 66% probability, although s/he only has 33% chance to discover Bob's real salary 1020. None of the existing anonymization principles (e.g., k -anonymity, l -diversity, etc.) can effectively prevent such proximity breach. The paper[1] remedied the problem by introducing a novel principle called (ε, m) -anonymity

2 Proximity Privacy and (ε, m) -anonymity

Let T be a microdata table that contains the private information of a set of individuals. T has d QI-attributes A_1, \dots, A_d , and a sensitive attribute(SA) S . We consider that S is numerical. For each tuple $t \in T$, $t[A_i]$ ($1 \leq i \leq d$) denotes its value on A_i , and $t[S]$ represents its SA value.

For a tuple $t \in T$, its private neighborhood $I(t)$ is a range in the domain of S that contains the sensitive value $t[S]$ of t , its $I(t)$ is an absolute ε -neighborhood if $I(t) = [t[S] - \varepsilon, t[S] + \varepsilon]$, where ε is any non-negative value[1].

Definition 1. Given a real value ε and an integer $m > 1$, a generalized equivalence

class G^* fulfills absolute (ε, m) -anonymity, if $\forall t \in G^*, P(t) < \frac{1}{m}$, where $P(t) = \frac{x}{|G^*|}$ is

the risk of proximity breach, x is the number of tuples in G^* whose sensitive values fall in $I(t)$ and $|G^*|$ the size of G . A generalized table T^* fulfills absolute (ε, m) -anonymity if all equivalence classes in T^* fulfill absolute (ε, m) -anonymity [1].

Definition 2 (Problem Definition). Given a table T and two integer ε, m , anonymize it to be T^* such that T^* is (ε, m) -anonymity and that T^* has the smallest number of stars.

3 The Hardness of Star Minimization

In the first sight, a simple reduction from l -diversity seems to establish the NP hardness of (ε, m) -anonymity. More specifically, given a table where the max size cover set is 1, that is $g=1$, the optimal (ε, m) -anonymity generalization is also the optimal l -diversity generalization if we set $m=l$. It is natural to wonder, in the more general case $g>1$, is the Problem of star minimization still NP-hard? Next, first assuming $m=3$,

Table 3. Illustration of Reduction

		D	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	S
From D ₁	{}	a	0	1	1	1	1	1	1
		b	2	0	2	0	0	2	2
		c	3	3	0	3	3	3	3
		d	4	4	4	4	4	0	4
From D ₂	{}	1	0	0	5+ ε				
		2	6+ ε	6+ ε	0	0	6+ ε	6+ ε	6+ ε
		3	7+ ε	7+ ε	7+ ε	7+ ε	0	7+ ε	7+ ε
		4	8+ ε	0	8+ ε				
From D ₃	{}	α	9+2 ε	9+2 ε	0	0	9+2 ε	9+2 ε	9+2 ε
		β	10+2 ε	0	10+2 ε				
		γ	11+2 ε	0	11+2 ε	11+2 ε	0	11+2 ε	11+2 ε
		δ	0	12+2 ε					

we establish the NP-hardness of star minimization, even when the size of maxi cover set g is $|T|/3$ in the microdata. Later, we will extend the analysis to any $m > 3$. Our derivation is based on a reduction from a classical NP-hard problem 3-dimensional matching ($3DM$).

Specifically, let D_1, D_2, D_3 be three dimensions with disjoint domains, and these domains are equally large: $|D_1| = |D_2| = |D_3| = n$. The input is a set S of $d \geq n$ distinct $3D$ points p_d in the space $D_1 \times D_2 \times D_3$. The goal of $3DM$ is to decide the existence of an $S_0 \subseteq S$ such that $|S_0| = n$ and no two points in S_0 share the same coordinate on any dimension. For example, assume $D_1 = \{a, b, c, d\}$, $D_2 = \{1, 2, 3, 4\}$, and $D_3 = \{\alpha, \beta, \gamma, \delta\}$, and a set S of 6 points are $\{p_1 = (a, 1, \delta), p_2 = (b, 1, \gamma), p_3 = (c, 2, \gamma), p_4 = (b, 2, \alpha), p_5 = (b, 3, \gamma), p_6 = (d, 4, \beta)\}$. Then, the result of $3DM$ is “yes”: a solution S_0 can be $\{p_1, p_3, p_5, p_6\}$.

Let v_1, \dots, v_n be the values in D_1 , v_{n+1}, \dots, v_{2n} be the values in D_2 , and v_{2n+1}, \dots, v_{3n} be the values in D_3 . We construct a microdata table T from S . Table 3 demonstrates the T built from the S , when $m = 3$.

Lemma 1. The $3DM$ on S returns yes, if and only if there is a $(\varepsilon, 3)$ anonymity generalization of T .

To extend the above analysis in a straightforward manner, we can show that, for any $m > 3$, optimal (ε, m) -anonymity under the constraint $m > 3$ is also NP-hard, through a reduction from m -dimensional matching.

4 Generalization Algorithm

In this section, we discuss how to develop a good partitioning strategy for suppression of (ε, m) -anonymity. The microdata T was first sorted in ascending order of their SA values, which was facilitating the step 6 of the partitioning algorithm to decide whether T_j satisfy (ε, m) -anonymity.

To reduce information loss, we will partition the microdata following the steps: we order the attributes according to their domain size; then the tuples in T are sorted in lexicographical QI-attribute order, based on the attribute ordering. First, we consider

attribute A_1 with the smallest domain size and put tuples with the same A_1 -value in the same partition. This results in a set of partitions $T_1, T_2, \dots, T|A_1|$, such that the number of A_1 will be 0 in all partitions (step 3-4). Then, for each sub-table T_i , we need to check whether T_i satisfy (ϵ, m) -anonymity, if yes, T_j was insert into the set S , and will participate the next round of partitioning. Otherwise, find a sub-table T_x such that (ϵ, m) -anonymity is satisfied by the merged partition $T_j \cup T_x$. If such a partition cannot be found, we merge T_j with a random partition and merging continues until the resulting partition satisfies (ϵ, m) -anonymity (Step 7).

In Step 8-9, let $g = \text{maxsize}(T)$, we divide each sub-table $T \in S$ into g disjoint QI-groups G_1, G_2, \dots, G_g by distributing tuples into G_i in a round-robin fashion: tuple $t_i (1 \leq i \leq n)$ is added to G_j , where $j = (i \bmod g) + 1$. In this way, for any two tuples $t_1, t_2 \in G$, $|t_1[S] - t_2[S]| > \epsilon$ and $|G| \geq m$.

The Partitioning Algorithm

Input: A microdata T , integers m and ϵ

Output: A set S consisting of sub-tables of T ;

Method:

1. $S = \{T\}$;
2. For $i=1$ to d
3. Pick the dimension A_i with minimized domain $|A_i|$;
4. Partition each $T \in S$ into $|A_i|$ sub-tables T_j ;
5. For $j=1$ to $|A_i|$
6. If T_j satisfy (ϵ, m) -anonymity, Add T_j to S ;
7. else merge with some partition T_k ;
8. For each sub-table $T \in S$ such that $|T| \geq 2m$;
9. Split the group into $|T|/\text{maxsize}(T)$ groups such that each group has at least m tuples;
10. Return S ;

5 Related Work

This section surveys the previous work on privacy preserving publication. The existing works can be loosely classified into two main categories. The first one aims at devising generalization principles, which serve as the criteria for judging whether a published relation provides sufficient privacy protection. Numerous generalization principles have been developed. These include k -anonymity [2–4], l -diversity [11], t -closeness [12] and so on. The second category includes algorithms for obtaining a generalized table under a generalization principle, which minimizes some quality metric. It has been shown [5] that computing the optimal generalized table is NP hard, even when a simple quality metric is deployed.

The existing generalization algorithms can be further divided into heuristic and theoretical. The main advantage of heuristic algorithms is that they are general, namely, they can be applied to many anonymization principles. Greedy solutions have also been proposed [6, 7, 9] to obtain a suboptimal solution much faster. Top-down and

bottom-up algorithms are presented in [6]. Incognito [8] borrows ideas from frequent item set mining, while Mondrian [9] takes a partitioning approach reminiscent of kd -trees. Authors of [13] are the first to establish the complexity of optimal k -anonymity, by showing that it is NP-hard to compute a k -anonymous table that contains the minimum number of stars. Xiao[14] present the first theoretical study on the complexity and approximation algorithms of l -diversity.

6 Empirical Evaluation

This section experimentally evaluates the effectiveness and efficiency of the proposed technique. Our purposes are twofold. First, we illustrate that our generalization algorithm (presented in Section 3) produces (ε, m) -anonymous tables that permit accurate data analysis. Second, we verify that the algorithm entails small computation cost. Our machine runs on 1.9 GHz AMD Core CPU, and has 1 gigabytes memory.

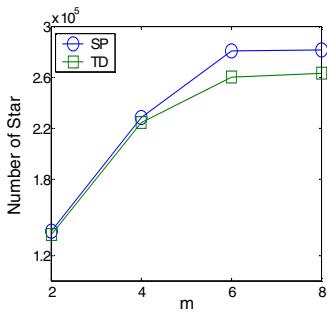


Fig. 1. Number of Stars vs. m

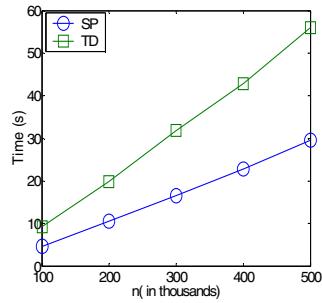


Fig. 2. Running Time vs. n

Our experimentation deploys a real database income frequently used in the literature, including the paper [1]. It contains $500k$ tuples, each of which describes the personal information of an American. Income includes nine integer attributes. In the following experiments, we compare our algorithm (denoted by TD) with the existing state-of-the-art technique: the splitting [1] algorithm(SP for short). We compare the usefulness of the data published by the two approaches, under the same privacy requirement.

The first set of experiments studies the influence of m on data utility. Towards this, we set ε to 3000. We measure the average (per-query) error of Tao's algorithm and our algorithm in answering a workload with $s = 0.1$ and first five columns as QI-attribute. Concerning absolute $(3000, m)$ -anonymity, Figure 1 plots the number of star as a function m . Both techniques incur more stars as m increases since a larger m demands stricter privacy preservation, which reduces data utility. Apparently, our algorithm produces more useful anonymized data than SP algorithm. Conversely, the number of star in SP algorithm deteriorates drastically as m increase to 8.

In the second experience, we evaluate the overhead of performing (ε, m) -anonymity. Assuming $\varepsilon = 3000$, figure 2 illustrates the cost of computing absolute

(ϵ, m) -anonymous generalization, when n varies from $100k$ to $500k$. The cost increases as n grows. This is expected, since more tuples that need to be anonymized will consume longer time to finish the anonymization procedure.

7 Conclusion

With the growing use of public services, privacy attracts increasing attentions from users, industry and the research community. In this paper, we present the first theoretical study on the complexity, and the corresponding algorithm to implement it. Through our experiments, we showed that our approximation algorithms perform significantly better than traditional approximation algorithms.

This work was supported by Project supported by the National Nature Science Foundation of China (No.61202007, 61033010, 61170006), and the Education Department of Zhejiang Province (Y201224678).

References

- [1] Li, J., Tao, Y., Xiao, X.: Preservation of Proximity Privacy in Publishing Numerical Sensitive Data. In: ACM SIGMOD 2008, Vancouver, BC, Canada (2008)
- [2] Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems 10(5), 557–570 (2002)
- [3] Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information. In: Proceedings of the 17th ACM Symposium on the Principle of Database Systems, Seattle, WA (June 1998)
- [4] Samarati, P.: Protecting respondents identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
- [5] Bayardo, R., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proc. of International Conference on Data Engineering (ICDE), pp. 217–228 (2005)
- [6] Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., Fu, A.W.-C.: Utility-based anonymization using local recoding. In: SIGKDD, pp. 785–790 (2006)
- [7] Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: ICDE, pp. 205–216 (2005)
- [8] LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proc. of ACM Management of Data (SIGMOD), pp. 49–60 (2005)
- [9] LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proc. of International Conference on Data Engineering (ICDE), pp. 277–286 (2006)
- [10] Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. In: ICDE, pp. 1–24 (2006)
- [11] Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE, pp. 106–115 (2007)
- [12] Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: PODS, pp. 223–228 (2004)
- [13] Xiao, X.K., Ke, Y., Tao, Y.F.: The Hardness and Approximation Algorithms for L-Diversity. In: EDBT 2010 (2010)

Annotating Web Images by Combining Label Set Relevance with Correlation^{*}

Feng Tian^{1,2,**} and Xukun Shen¹

¹ State Key Laboratory of Virtual Reality Technology and Systems,
Beihang University, 100191 Beijing, China

² School of Computer and Information Technology,
Northeast Petroleum University, 163318 Daqing, China
{tianfeng,xkshen}@vrlab.buaa.edu.cn

Abstract. Image annotation can significantly facilitate web image search and organization. Although it has been studied for years by the computer vision and machine learning communities, image annotation is still far from practical. Existing example-based methods are usually developed based on label co-occurrence information. However, due to the neglect of the associated label set's internal correlation and relevance to image, the annotation results of previous methods often suffer from the problem of label ambiguity and noise, which limits the effectiveness of these labels in search and other applications. To solve the above problems, a novel model-free web image annotation approach is proposed in this paper, which consider both the relevance and correlation of the assigned label set. First, measures that can estimate the label set relevance and internal correlation are designed. Then, according to the above calculations, both factors are formulated into an optimization framework, and a search algorithm is proposed to find a label set as the final result, which reaches a reasonable trade-off between the relevance and internal correlation. Experimental results on benchmark web image data set show the effectiveness and efficiency of the proposed algorithm.

Keywords: label set relevance, web image annotation, image label.

1 Introduction

Automatic image annotation, which enables conversion of image retrieval into text matching, has received much research interest recently. These approaches can be divided into the following two types: learning based approaches and example-based approaches. The learning-based approaches such as cross-media relevance model(CMRM), generative model based on continuous features (CRM) learn the mapping model between visual features and semantic concepts[1]. Models such as latent dirichlet allocation, probabilistic latent semantic analysis, and

* This work is supported by Scientific Research Fund of Heilongjiang Provincial Education Department(NO:12511011,12521055).

** Corresponding author.

hierarchical dirichlet processes annotated images as samples from a specific mix of topics, where each topic is a distribution over image features and annotation words. Discriminative models such as multiple-label learning, multiple-instance learning learn classifier for labels. All of above methods showed their considerable performance on small scale image and vocabulary set, whereas the complexity of the learned model and their poor scalability make the learned model hardly generalizable and unrealistic to application on infinite semantic concepts set for large scale web image annotation. Different from the first group, example-based algorithms assume that similar image share similar label[2][3][4][5]. These example-based methods needn't parameters learning and parameter estimation, and are data-driven and model-less which are more flexible and can be applicable to large scale data set. In [3], content-based image retrieval is used to retrieve a set of visually similar images from a large-scale web image set, text-based label search is used to obtain a ranked list of candidate annotations for each retrieved image, then the top ones in the ranked label lists are annotated. In [4], initial relevance scores for the labels is estimate based on probability density estimation, and a random walk over a label similarity graph is performed to refine the relevance scores. In [5], the author proposes a neighbor voting algorithm to learn label relevance. However, this type of approach's performance is inferior relatively. Recent studies indicates that the correlation between the labels can improve learning quality[6][7][8]. But these methods often rely on complicated learning algorithms and are not easy to model the correlations between labels when extending to large number of labels. Intuitively, such information is helpful for us to better understand image content. The ideal label set associated with images should be relevant to the image, namely, the label set can accurately describe the content of the image; Meanwhile, the ideal label set should be internal correlative,namely, labels in the set is correlative to each other. Label set's relevance to image and correlation between a label and label set are quite helpful to better understand image semantic, meanwhile, large scale web image annotation require the learning method has both effectiveness and efficiency. So we formulate an optimization framework that includes both factors and solve the problem in a heuristic way.

2 Problem Formulation

Given an image collection X , a vocabulary W , we define a candidate label set W_q with q labels, the relevance of W_q to image x is defined as $r(x, W_q)$, and the correlation between labels in W_q with respect to x , namely, internal correlation of W_q is defined as $c(x, W_q)$. The objective is to assign the most relevant and internal correlative label set to x . We use $F(x, W_q)$ to denote the objective. Thus, the optimization problem is:

$$\hat{W}_q = \underset{W_q \subset W}{\operatorname{argmax}} F(x, W_q) = \operatorname{argmax} (\alpha \cdot r(x, W_q) + (1 - \alpha) \cdot c(x, W_q))$$

where \hat{W}_q is the ideal label set to x , α is a controlling parameter. It is unrealistic to solve this problem by greedy search when the whole vocabulary size n

is very large, so we propose a heuristic iterative algorithm to solve it and get an approximate optimal solution, that is, searching the label which is most relevant to x and correlative to W_{q-1} at q-th iteration. Given a label denoted as w , we have $r(x, W_q) \approx r(x, W_{q-1}) + r(x, w)$, where $W_q = W_{q-1} \cup \{w\}$, $r(x, w)$ denotes the relevance of w to x . Similarly, the internal correlation of W_q with respect to image x can be estimated by $c(x, W_q) \approx c(x, W_{q-1}) + c(W_{q-1}, w)$, where $c(x, W_{q-1})$ denotes the internal correlation of W_{q-1} , and $c(W_{q-1}, w)$ denotes the relevance of label w to W_{q-1} with respect to image x . Then, we select the label which maximizes $F_w(x, w)$ at q-th iteration and added it into W_{q-1} , where $F_w(x, w)$ integrates the label w 's relevant and correlated information added to W_{q-1} . Thus, we have

$$\hat{w} = \underset{w \in W/W_q}{\operatorname{argmax}} F_w(x, w) = \underset{w \in W/W_{q-1}}{\operatorname{argmax}} (\alpha \cdot r(x, w) + (1 - \alpha) \cdot c(W_{q-1}, w)) \quad (1)$$

3 "Label Set"-to-Image Relevance

"Label set"-to-image relevance indicates label set's efficiency in describing the image visual content. There are two cases will result in label set's ambiguous description of image. The first case is polysemy. Considering the following scenario, label set "apple", "photo" can describe a fruit or a computer, while "apple", "computer" or "apple", "fruit" can be more discriminant, so the latter two label sets are more relevant to the target image. The second case is that label set is not specific. For example, a label set "car" is not specific since there are various brand like "Chevrolet" or "Volkswagen". Labels "nice" or "cool" added into will not give more relevant information to the image. Note that, from the perspective of the whole label set, the meaning of a label's relevance to one image indicates how much relevant information it added to the label set, which can be reflected from the ambiguity changes. Intuitively, adding one informative label to the label set, will result in great changes to the posterior distributions of other labels. Then, the degree of posterior distributions of other labels changed after this label joined into the label set can estimate the relevant information of a label. For example, "yellow", "Chevrolet", "Volkswagen", "Germany", and "US" are likely to occur with the label set "car". However, labels Volkswagen and Germany are less likely to occur with the updated label set "car", "Chevrolet" when the label Chevrolet is added into the label set "car", thus their posterior distributions to the labels in the set will be changed. This means that the label set "car", "Chevrolet" is more specific than the label set "car", and has more relevant information. Thus, when a label w is added in, we can compute the K-L divergence of the posterior distributions $P(w'|W_{q-1})$ and $P(w'|W_q)$, where $w' \in W$. The greater the divergence value is, more chance w is selected into the label set. Thus, we have

$$\begin{aligned} r(x, w) &\propto f(KL(P(W|W_{q-1} \cup \{w\})||P(W|W_{q-1}))) \\ &= f(KL(P(W|W_q)||P(W|W_{q-1}))) \end{aligned} \quad (2)$$

where $f(\cdot)$ should be a monotonically increasing function, and K-L divergence can be computed by:

$$KL(P(W|W_q) || P(W|W_{q-1})) = \sum_{w' \in W} P(w'|W_q) \log \frac{P(w'|W_q)}{P(w'|W_{q-1})}$$

Here

$$P(w'|\Omega) = \frac{P(\Omega|w')P(w')}{P(\Omega)} = \frac{P(w') \prod_{w_1 \in \Omega} P(w_1|w')}{\sum_{w_2 \in W} P(w_2) \prod_{w_1 \in \Omega} P(w_1|w_2)}$$

, where Ω denotes W_q or W_{q-1} . The label's conditional probability can be easily computed by labels co-occurrence. Since there have large differences in the frequency of different labels, estimating their prior probability directly by their frequency simply to will include bias, so we compute conditional probability and prior probability as:

$$P(w_i|w_j) = \frac{\text{count}(w_i, w_j)}{\sum_{k=1}^q \text{count}(w_j, w_k)}$$

,

$$P(w_i) = \frac{\sum_{j=1}^q \text{count}(w_i, w_j)}{\sum_{j=1}^q \sum_{k=1}^q \text{count}(w_j, w_k)}$$

where $\text{count}(a, b)$ denotes the co-occurrence number of a and b . By Formula(2), We select the label by the vocabulary's KL divergence of posterior distribution only. However, relevance between labels and image is not considered. For example, given a candidate label set "car", "coffee" may be selected to added into the set since it leads to the greater change of posterior distribution of labels in the vocabulary. So formula(2) is rewritten as:

$$r(x, w) \propto f(KL(P(W|W_{q-1} \cup \{w\}) || P(W|W_{q-1}))) \cdot P(w|x)$$

Assume $P(x)$ is uniformly distributed, thus $P(w|x) \propto P(w, x)$

$$r(x, w) \propto f(KL(P(W|W_{q-1} \cup \{w\}) || P(W|W_{q-1}))) \cdot P(w, x) \quad (3)$$

where $P(w, x)$ can be estimated with the expectation over the nearest neighbors of x as follows,

$$\begin{aligned} P(w, x) &= \sum_{i=1}^k P(w, x|x_i)P(x_i) = \sum_{i=1}^k P(w|x_i)P(x|x_i)P(x_i) \\ &\approx \frac{1}{k} \sum_{i=1}^k \delta(w, x_i) \cdot sim_{visual}(x, x_i) \end{aligned} \quad (4)$$

Here, $\delta(w, x_i)$ denotes whether x_i has label w , $sim_{visual}(x, x_i)$ is the visual distance of samples. Note that, we should consider the label similarity of images in

selecting neighbors, so we retrieve K ($K > k$) neighbors by visual similarities, then pair wise label vector's similarity is computed as

$$\text{sim}(x_i, x) = \frac{1}{K} \sum_{x_j \in N(x)} \text{sim}_{\text{text}}(x_i, x_j) \cdot \text{sim}_{\text{visual}}(x, x_j) \quad (5)$$

where $\text{sim}_{\text{text}}(x_i, x_j)$ is the similarity of corresponding label vector, which can be computed by Cosine distance, after that, top-k samples with great $\text{sim}(x_i, x)$ is selected.

4 Label Set Internal Correlation

We assumed labels are independent in the estimating label set relevance in section 3. However, it may results in deviation because it does not consider the internal correlation of label set. Considering the following scenario, a label set "sky", "sun" has been assigned to image, then the label "rain" may be added in since it has correlation with "sky", but "rain" has little relevance with "sky", "sun" as a whole. So we claimed that the label set should be internal correlative as a whole, namely, each label should be relevant to all other labels in the label set. Meanwhile, the label which is less relevant to the image but relevant to the label set should be added in. The $c(W_{q-1}, w)$ defined in Section 2 is proposed to measure the relevance between the candidate label set W_{q-1} and label w . Thus, we have $c(W_{q-1}, w) = \text{sim}_{\text{text}}(W_c, w)$, here both the candidate label set W_{q-1} and label w are represented by vector, then we get the "label set vector" W_c by a label combination process that combine all the labels in the candidate label set. $\text{sim}_{\text{text}}(W_c, w)$ is the semantic similarity between W_c and w . To get the "label set vector" W_c , the vector of each label need to be constructed first. Assuming that each image be a document containing the associated labels, we can use the following method to measure the semantic relevance between labels. First, each image is represented by its associated labels. Then, the label-to-image matrix that describes the relationship between labels and image is denoted as $M_{|W| \times n}$, where $|W|$ is the vocabulary size, n is the number of images in the training set. The i-th row of M denotes the occurring pattern of label w_i in the training set, and the j-th column of M denotes the labeling of image x_j , $M_{i,j}$ denotes whether w_i is associated with x_j . Then, the label-to-label matrix $U_{|W| \times |W|} = MM^T$ is obtained, which describes labels' semantic correlation. We normalize the matrix U by $U_{ij} = \frac{U_{ij}}{U_{ii} + U_{jj} - U_{ij}}$ since there are variations in the frequency of different labels, and U_{ij} denotes co-occurrence information of w_i and w_j . Thus, the U 's i-th row vector U_i can be regarded as the neighborhood vector of w_i , and the semantic correlation of w_i and w_j can be estimated by their neighborhood vector U_i and U_j . Here, we construct W_c by a heuristic label combination process. Given two label's neighborhood vector $\vec{w}_i = \langle v_{i1}, \dots, v_{i|W|} \rangle = \langle \beta_i U_{i,1}, \dots, \beta_i U_{i,|W|} \rangle$ and $\vec{w}_j = \langle v_{j1}, \dots, v_{j|W|} \rangle = \langle \beta_j U_{j,1}, \dots, \beta_j U_{j,|W|} \rangle$, where β_i represent the label's weight w_i to x and is computed by $\beta_i = \frac{|D_{neighborhood}(w_i)|}{\log(|T(w_i)|+1)} \cdot p(w_i, x)$ where $|D_{neighborhood}(w_i)|$ denotes the number of images that contain label w_i in the

K nearest neighbours of the test image x , $|T(w_i)|$ denotes the number of images that contain label w_i in the training dataset. The result combined label vector is

$$v_{w_i \cup w_j, k} = \begin{cases} \frac{\max(\beta_i, \beta_j)}{\beta_j} (v_{ik} + v_{jk}) & v_{ik} \cdot v_{jk} \neq 0 \\ v_{ik} + v_{jk} & \text{else} \end{cases} \quad (6)$$

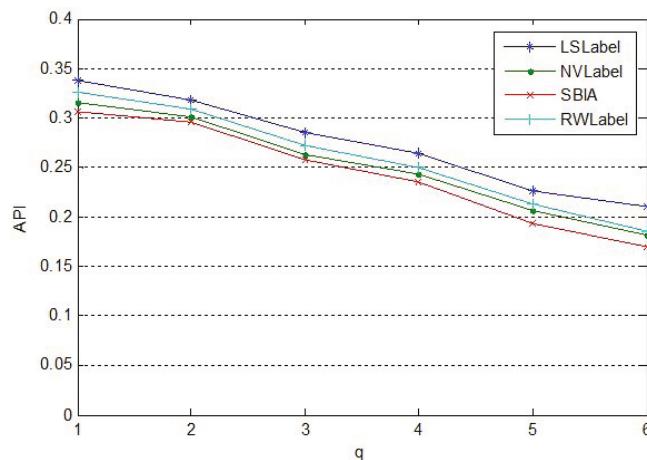
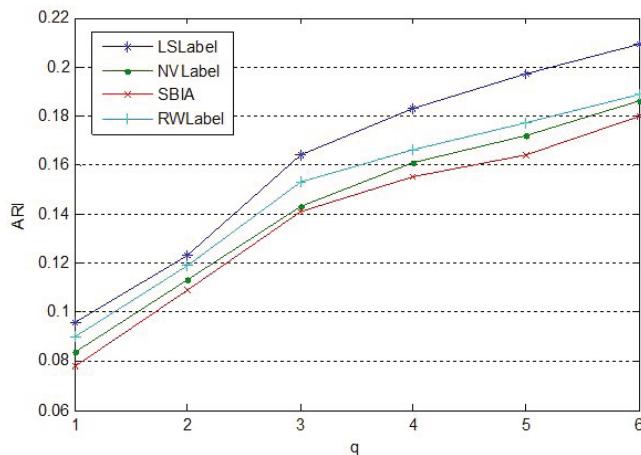
where $\max(\beta_i, \beta_j)$ is an augment factor. Thus the "label set vector" W_c is a combination of $(q-1) \times 1$ "neighborhood vector" by $(\dots((w_1 \cup w_2) \cup w_3) \cup \dots \cup w_{q-1})$. In the annotation process, we combine the concept W_c and the candidate label w_q which is selected to join in label set W_{q-1} by the label combination process : $\bigcup_{w_i \in W_q} w_i = (\bigcup_{w_i \in W_{q-1}} w_i) \cup w_q$. Therefore, the semantic similarity of "label set vector" W_c and label w_i can be computed by $\text{sim}_{text}(W_c, w_i) = \frac{W_c \cdot U_i}{|W_c| \times |U_i|}$

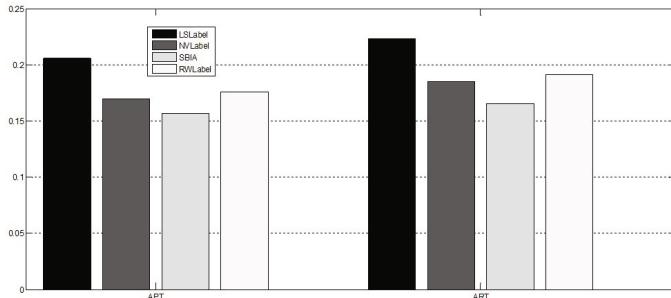
5 Experiments

We conduct extensive experiments on dataset NUS-WIDE[9] which contains 269,648 images and 5,018 labels all of which exist in WorldNet. We split NUS-WIDE into a training set with 150,000 images and a test set with the remaining images. We use two types of global image descriptors: Gist features, and color histograms with 16 bins in each color channel for RGB and HSV representations. Local feature SIFT are extracted densely on a multiscale grid or for Harris-Laplacian interest points. Images are then represented as a bag-of-words histogram. We concatenate them to form a new global descriptor. We evaluate our models with standard performance measures. Let n_i^s be the number of images automatically annotated with a given label, n_i^p be the number of images correctly annotated with that label. n_i is the number of images having that label in ground-truth. Thus $ARL = \sum_{i=1}^{|W|} \frac{n_i^p}{n_i}$, $APL = \sum_{i=1}^{|W|} \frac{n_i^p}{n_i^s}$, $F1 = \frac{2 \times ARL \times APL}{ARL + APL}$. Average precision $API@q$ and recall $ARI@q$ of image with q result labels are also used to evaluate the method: $API@q = \frac{1}{|T|} \sum_{i=1}^{|T|} \frac{|P(I_i)|}{q}$, $ARI@q = \frac{1}{|T|} \sum_{i=1}^{|T|} \frac{|P(I_i)|}{|G(I_i)|}$, where T is the test image set, $P(I_i)$ is the set of result labels correctly assigned to image I_i by the algorithm, $G(I_i)$ is the set of ground truth labels for image I_i . We compare our algorithm LSLLabel with the following state of the art web image annotation algorithms, i.e., SBIA(Search based Method)[3], RWLabel [4] and NVLabel [5]. Table 1 shows the API and ARI with q varied from 1 to 6. Figure 1 and Figure 2 shows the corresponding curve for comparison. According to the results, LSLLabel archives encouraging improvements, and the API and ARI are greatly improved. From these figures, we can observed that, our proposed algorithm LSLLabel outperforms other algorithm significantly. For example, when q is fixed as 6, LSLLabel has an improvement about 16% over NVLabel, 24% over SBIA, 14% over RWLabel in terms of API. In terms of ARI it has an improvement about 12% over NVLabel, 16% over SBIA, 11% over RWLabel. The reasons are that, it is difficult for SBIA to properly determine the number of

Table 1. Annotation performance with q varied from 1 to 6 on NUS-WIDE dataset

q	ls_{API}	ls_{ARI}	nv_{API}	nv_{ARI}	$sbia_{API}$	$sbia_{ARI}$	rw_{API}	rw_{ARI}
1	0.338	0.096	0.315	0.084	0.306	0.078	0.326	0.090
2	0.318	0.123	0.301	0.113	0.296	0.109	0.309	0.119
3	0.285	0.164	0.263	0.143	0.258	0.141	0.272	0.153
4	0.264	0.183	0.243	0.161	0.236	0.155	0.250	0.166
5	0.226	0.197	0.207	0.172	0.194	0.164	0.213	0.177
6	0.211	0.209	0.182	0.186	0.170	0.180	0.186	0.189

**Fig. 1.** Curve of API with q varied from 1 to 6**Fig. 2.** Curve of ARI with q varied from 1 to 6

**Fig. 3.** Comparison of APL & ARL

clusters. NVLabel usually assign common labels which has larger frequency in the neighborhood. This is the same to RWLabel in which the labels has the most correlation to other labels are assigned to the test image. However, the LSSLabel prefers the relevant label set as a whole. With assigned label's number q increasing the superiority is more obvious. To compare the average precision and recall in terms of label, we fix the number of labels for image as 6 and obtain the APL and ARL. Figure 3 shows the results. From these figures, we can

Table 2. Annotation results generated by different methods (labels in italic font are matching)

	ferrari automobile car	apple red splash water	storm rain sky clouds
Manual Labels	red fast concept grass	fruit food nikon	bike bay asia
	green italy sports	august art droplets	people kodak baywalk
	<i>ferrari car grass italy</i>	<i>apple fruit leaf tree</i>	<i>sky rain storm bike</i>
LSSLabel	model sports luxury	food table red	mountain clouds people
	tourist green road	sour green juicy	water tree silhouette
	<i>car automobile ferrari</i>	<i>apple fruit cake summer</i>	<i>sky sun hot clouds</i>
RWLabel	police grass BMW girl	computer pears red	tree bay bike
	BENZ tree speed	EOS tree square	people ocean bank
	<i>car people ferrari police</i>	<i>apple leaf people</i>	<i>sky sun airplane clouds</i>
SBIA	grass model tourist	beauty green fruit nice	tree silhouette bike
	fast KIA sports	cup man photo	people cannon
	<i>car auto automobile light</i>	<i>apple tree leaf table</i>	<i>clouds sky sun storm</i>
NVlabel	BENZ ferrari girl	green red fruit	airplane boat bay
	road luxury grass	computer autumn cake	bridge people happy

observed that for the average precision, LSLLabel has an improvement 17-31% over other annotation algorithms, and for the average recall of label, LSLLabel has an improvement about 17-35% over the other annotation algorithms. Because these algorithms prefer the labels which appear frequently in the visually similar images. However, the LSLLabel prefers the relevant label set as a whole, and the label set which is correlative and has great relevance information to image is assigned. Table 2 shows some annotation results generated by different methods on NUS-WIDE dataset. It shows that label set obtained by our algorithm is more discriminant to describe corresponding image. Our algorithm selects the label "ferrari" for the first image, because it disambiguates the image from "Benz", "BMW" and "KIA". RW method and Neighbor Voting method also select label "BENZ", "BMW" or "KIA". This also happens to the second image, our algorithm selects the relevant label "fruit", and it can disambiguate the image from computer. However, other algorithms also select the label "computer" because label "computer" has high co-occurrence frequency with "apple". Meanwhile, other algorithms label the third image with labels "sun" and "hot", which is ambiguous to describe the third image.

Figure 4 shows examples of F1 scores for some individual labels. We can also observed that LSLLabel get best performance in comparison with other methods.

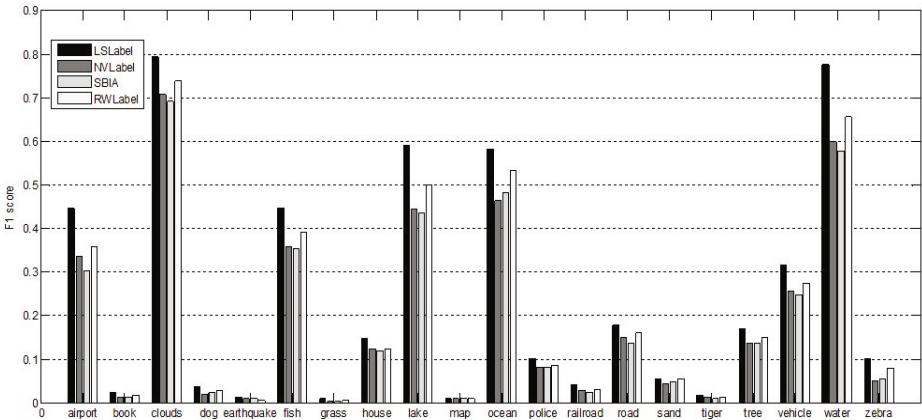


Fig. 4. Comparison of F1 for some labels

6 Conclusions

In this paper, a new model for web image annotation is proposed by simultaneously learning "label set"-to-image relevance and label set internal correlation in a joint framework. The "label set"-to-image relevance is computed based on the posterior probability computation using KL divergence and label set's internal

correlation is computed by label correlation analysis. We further solve the label set selection problem in a heuristic and iterative manner, thus making the framework more applicable to the large scale annotation environment. Experiments show that the proposed method achieves excellent and superior performance.

References

1. Makadia, A., Pavlovic, V., Kumar, S.: A new baseline for image annotation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part III*. LNCS, vol. 5304, pp. 316–329. Springer, Heidelberg (2008)
2. Wang, X.: AnnoSearch: Image Auto-Annotation by Search. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1483–1490. IEEE Computer Society Press, Los Alamitos (2006)
3. Wang, X.-J.: Annotating images by mining image search results. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 1919–1932 (2008)
4. Liu, D., Hua, X.-S., Yang, L.: Tag Ranking. In: *ACM International Conference of World Wide Web*, pp. 351–360. ACM Press, New York (2009)
5. Li, X.: Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia* 11, 1310–1322 (2009)
6. Kang, F.: Correlated label propagation with application to multi-label learning. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1719–1726. IEEE Computer Society Press, Los Alamitos (2006)
7. Wang, H., Huang, H., Ding, C.H.Q.: Image annotation using multi-label correlated Greens function. In: *International Conference on Computer Vision*, pp. 1–8. IEEE Computer Society Press, Los Alamitos (2009)
8. Wang, H., Huang, H., Ding, C.: Multi-label Feature Transform for Image Classifications. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) *ECCV 2010, Part IV*. LNCS, vol. 6314, pp. 793–806. Springer, Heidelberg (2010)
9. Chua, T.-S., Tang, J., Hong, R.: NUS-WIDE: A real-world web image database from National University of Singapore. In: *ACM International Conference on Image and Video Retrieval*, pp. 1–9. ACM Press, New York (2009)

Clustering Analysis and Semantics Annotation of 3D Models Based on Users' Implicit Feedbacks

Lv Tianyang^{1,2,3,*}, Huang Shaobin¹, Wu Peng¹, and Lang Dapeng¹

¹ College of Computer Science and Technology,
Harbin Engineering University, Harbin, China

² Audit Research Institute, National Audit Office, Beijing, China

³ Computer Technology Center, National Audit Office, Beijing, China

raynor1979@163.com

Abstract. The performance of 3D model retrieval can be greatly improved by adopting precise semantics. Since manual annotation of semantics is too time-consuming, it is necessary to explore an automatic or semi-automatic way. Although it is widely accepted that users' feedbacks contain semantics, previous researches usually utilize relevance feedbacks in computing similarity of 3D models. The paper proposes a strategy for semantics clustering, annotation and retrieval of 3D models, which adopts not only relevance feedbacks but also noisy user operations. The strategy first converts implicit feedbacks into a weighted semantics network of 3D models. After analyzing this semantics network, this paper proposes an agglomerative hierarchical clustering method based on a novel concept of semantics core to obtain the semantics communities under different granularity. Finally, this paper shows an automatic semantics annotation method using the semantics of only a few 3D models. The proposed method is verified by simulated feedbacks with strong noise and real feedbacks of the Princeton Shape Benchmark. Our experiments show that the strategy achieve good performance not only in semantics clustering but also in semantics annotation.

Keywords: 3D model retrieval, implicit feedback, complex network, semantics clustering, semantics annotation.

1 Introduction

With the proliferation of 3D models, 3D model retrieval emerges as an important research field in the last decade. The performance of 3D model retrieval can be greatly improved by utilizing semantics[11]. It is well accepted that the semantics of an object is related to human's understanding. And the semantics is generally represented by the keywords at present.

Compared to the document retrieval, 3D model retrieval lacks valuable text information. While it is too expensive for experts to annotate semantics of a large amount of 3D models, it encounters quality problem to extract semantics from

supplemental texts like saving path or file name of a 3D model. Therefore, it becomes an important topic to acquire semantics of 3D models semi-automatically or automatically.

Since feedbacks reflect users' understanding of the retrieved objects, they contain semantics. User feedback can be classified into two categories: (1) implicit feedback or operational information, such as mouse click; (2) relevance feedback, which can be further divided into the positive and the negative feedback.

Previous researches usually utilize relevance feedback in similarity computing [1], especially in image retrieval. And a few works also adopt relevance feedback in semantics annotation [3]. Since users is unwilling to provide relevance feedback, ref. [5] classifies the queries based on user's clicks.

However, only a few researches focus on 3D model retrieval. And the relationship between implicit feedback and semantics of 3D model still needs to be explored. It is not an easy task, because users' operations vary differently and contain lots of noises.

This paper proposes a strategy novel based on complex network to analyze semantics clusters and to annotate semantics of 3D models according to noisy implicit feedbacks.

This paper first constructs a weighted semantics network of 3D models according to implicit feedbacks based on our previous work[4]. Thus, the semantics of 3D models is reflected by their topology and clustering property in the complex network and will be represented by several keywords after semantics annotation. Then, the paper proposes a robust agglomerative hierarchical clustering algorithm of complex network to discover semantics communities with different granularity. The algorithm is named CACNSC. Finally, based on the hierarchical structure of semantics relationship, the semantics of all 3D models can be automatically annotated according to that of a few models whose semantics can be already existed or decided manually. The proposed strategy requests no additional work from users.

In addition to the simulated dataset with lots of noise, the proposed method is verified by the real feedbacks of Princeton Shape Benchmark[8] (termed PSB in the following parts). PSB is a benchmark of 3D model retrieval. Serial experiments show that the proposed method performs quite well not only in semantic clustering but also in annotation.

2 Construction of Weighted Semantics Network Based on Implicit Feedback

As stated in our previous work[4], there are two phenomena coexisting in implicit feedbacks: the random of single feedback and the commonness of lots of feedbacks. Therefore, high quality semantics can be revealed by utilizing lots of feedbacks. Our preliminary research has verified this assumption.

This section introduces the weighted complex network to express semantics relationships among 3D models, which are constructed based on implicit feedbacks. A feedback can be a user operation or a positive feedback.

Suppose there is a multimedia database $O = \{o_1, o_2, \dots, o_{N-1}\}$ with N 3D models. The database is categorized into l semantics classes $\{A_1, A_2, \dots, A_l\}$. These classes contain n_1, n_2, \dots, n_l 3D models respectively. Let q_r be the r -th query, where q_r can be a set of keywords, a sample 3D model etc.. Terming the feedback of q_r as (q_r, V_r) , where V_r is a N dimensional vector; $V_r^i = 1$, iff o_i appears in the feedback of q_r , otherwise $V_r^i = 0$. And all N_{fb} feedbacks are recorded as a $N \times N_{fb}$ matrix M_{fb} . Fig. 1.(b) shows an example of M_{fb} , which is extracted from 5 feedbacks of Fig. 1.(a) with $O = \{o_0, o_1, o_2, o_3\}$. And $(q_1, (1, 1, 0, 1))$ means that 3D models o_0, o_1 and o_3 appear in the feedback of q_1 .

We define the semantics relationship reflected by implicit feedbacks as follows:

Definition 1. *The semantics of o_i and o_j are related, if $V_r^i \times V_r^j = 1$ and $i \neq j$.*

The related times of o_i and o_j in N_{fb} feedbacks are termed as $sem(o_i, o_j)$. Thus, the degree of semantics relationship can be reflected by the value of $sem(o_i, o_j)$.

Therefore, we define the semantics relationship matrix M_{sem} among 3D models as follows:

Definition 2. *Define the semantics relationship matrix as $M_{sem} = [a_{ij}]_{N \times N}$, where $a_{ij} = sem(o_i, o_j)$ if $i \neq j$, otherwise $a_{ij} = 0$.*

Fig. 1.(c) shows the M_{sem} obtained from the feedbacks of Fig. 1.(b). since o_0 and o_1 appear in q_1 and q_4 , $sem(o_0, o_1) = 2$.

The matrix M_{sem} can be easily converted to a weighted complex network G_{sem} . Its node v_i corresponding to a 3D model o_i , the edge $e_{(i,j)}$ is the edge between v_i and v_j , the weight of $e_{(i,j)}$ is $w_{(i,j)}$ and $w_{(i,j)} = a_{ij} = sem(o_i, o_j)$. Other notations of a complex network[9] will also be used in the following parts: the degree of node v_i $d_{(v_i)}$, the strength of v_i S_{v_i} that equals the weight sum of the edges connecting to v_i and the average weight $\overline{w_{(v_i)}}$ of the edges connected to v_i .

Therefore, the analysis of implicit feedbacks is converted to the analysis of semantics complex network G_{sem} . The construction of G_{sem} needs no text information. And G_{sem} evolves with the accumulation of feedbacks.

3 A Hierarchical Clustering Algorithm of Complex Network Based on Semantic Core

It is an unsupervised learning problem to discover semantics clusters in G_{sem} without any prior knowledge of $\{A_1, A_2, \dots, A_l\}$. A cluster in a complex network is also termed as a *community* or a *group*. Ref. [7] shows that there is a hierarchical structure in many complex networks. Although many clustering algorithms have been proposed, their performances are still unstable for different problems.

This section introduces a robust hierarchical agglomerative clustering algorithm based on semantics core for analyzing 3D models semantics. The algorithm is designed to handle weighted networks with different local topologies. Based on the hierarchical clustering procedure, a binary tree reflecting semantics accumulation under different granularity can be obtained in section 4.

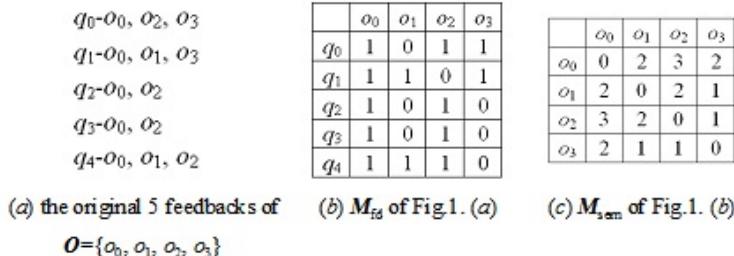


Fig. 1. The semantics relationship matrix extracted from implicit feedbacks

3.1 Characteristics of Weighted Semantic Complex Network

Previous researches on complex network clustering concentrate on analyzing non-weighted network. However, there are apparent differences between a non-weighted community and a weighted community. For instance, it is assumed that the connectivity of a intra-community is higher than that of inter-communities [10]. This assumption is not necessarily satisfied in a weighted network[6]. Fig.2(a) is an example, where the round nodes and the rectangle nodes form two communities. But the inter-connectivity between them is not lower than the intra-connectivity in any community, if the edge weight is not taken into consideration.

Meanwhile, a weighted semantics network has the following features:

- (1) Local topologies such as the size of communities are affected by the size of semantics classes, retrieval interest and noise. For instance, in PSB, the largest class "human" is 40 times bigger than the smallest class "ant". And people's searching interest will eventually decide the number of feedbacks related to a specific semantics class.
- (2) Nodes/models in the weighted network have quite different topology properties. A node with high $\overline{w}_{(v_i)}$ has much closer semantics relationships with some models. v_2 of Fig.2(a) is an example that connects very tightly with v_7 and v_{10} .

According to these features, the clustering method should take the local topology into consideration. Otherwise, small communities existing besides big ones, or a loose but big community neighboring to the compacted ones, won't be recognized. And the special nodes should have the priority to be detected.

3.2 The Definition and Detection of Semantic Core

Since the node with high $\overline{w}_{(v_i)}$ is closely related with some nodes, we define the set of these nodes as a semantics core.

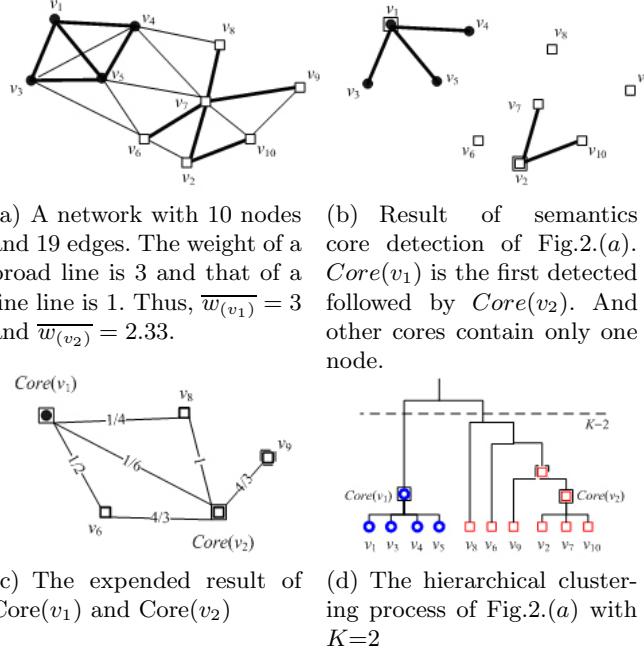


Fig. 2. Example of the clustering procedure of a network with 2 semantics classes

Definition 3. *Semantics core $Core(v_i)$ is a set of nodes v_i and all v_j that satisfy $v_j \in N_G(v_i)$ and $w_{(i,j)} \geq \max_{v_k \in \overline{N_G(v_i)}}(\overline{w}_{(v_k)})$.*

Where $N_G(v_i)$ is the set of nodes connected with v_i and $\overline{N_G(v_i)} = N_G(v_i) \cup \{v_i\}$. And v_i is called the centroid of $Core(v_i)$.

It is easy to prove that $\forall i, j, Core(v_i) \cap Core(v_j) = \emptyset$. According to the definition, the connectivity degree of a semantics core evaluated by \overline{w} is higher than that of its neighbors. This is a local constraint to adapt different local topologies in semantics network. Fig.2.(b) shows the semantics cores detected from Fig.2.(a). It shows that although the connectivity in a core is higher than that of its neighbors, it isn't the globally highest, such as $Core(v_2)$.

The algorithm Core_Detection detecting semantics core is listed as follows. After core detection, several small clusters are obtained before further hierarchical clustering. To further reduce the hierarchical clustering times, the detected semantics cores are expanded. first, G_{sem} is replaced by $G_{sem-core}$ by treating each semantics core as a single node. Then, the weights of edges between semantics cores $Core(v_i)$ and $Core(v_j)$ are updated according to formula (1). Finally, we perform Core_Detection($G_{sem-core}$) to expand cores.

$$w_{(Core(v_i), Core(v_j))} = \frac{\sum_{v_p \in Core(v_i)} \sum_{v_q \in Core(v_j)} w_{(p,q)}}{|Core(v_i)| \times |Core(v_j)|} \quad (1)$$

Algorithm 1. Core_Detection(G_{sem})

-
- 1: All nodes of V_{node} are sorted in descending order according to \overline{w} as $\{v_0, v_1, \dots, v_{N-1}\}$, and let $C = \emptyset$, $i = 0$;
 - 2: Obtain the semantics core $Core(v_i)$ of v_i ;
 - 3: **if** $|Core(v_i)| > 1$ **then**
 - 4: $V_{node} = V_{node} - Core(v_i)$, $C = C \cup Core(v_i)$
 - 5: **end if**
 - 6: **if** $i < N$ **then**
 - 7: $i = i + 1$, goto step 2, otherwise goto step 9
 - 8: **end if**
 - 9: Insert each remain node of V_{node} into C as a semantics core since one must be equivalent to itself in semantics.
-

3.3 Similarity Computation between Communities

Similarity computation is the key of clustering. To evaluate the local feature of a community C_i , define the density $den(C_i)$ as the average weight of all edges of C_i .

$$den(C_i) = \frac{\sum_{v_p \in C_i} \overline{w(v_p)}}{2 \times |C_i|} \quad (2)$$

Where $|C_i|$ equals the number of 3D models belong to C_i and $den(C_i)$ shows the relationship between the edges and their weights in C_i .

In the process of hierarchical clustering, the connectivity between C_i and C_j should be one of the highest if they are the candidates to be merged, while the local topology of C_i and C_j and that of the inter-part between C_i and C_j should be similar. In this case, the new community created by merging C_i and C_j can have similar intra-topology.

Define the factor that reflects the difference between local topology of C_i and that of the inter-part between C_i and C_j as follows:

$$\nabla_{den}(C_i) = \begin{cases} \frac{\overline{w}_{(C_i, C_j)}}{\overline{den}(C_i)}, & \text{if } \overline{w}_{(C_i, C_j)} \leq den(C_i)) \\ \frac{\overline{den}(C_i)}{\overline{w}_{(C_i, C_j)}}, & \text{if } \overline{w}_{(C_i, C_j)} > den(C_i)) \end{cases} \quad (3)$$

Where $\overline{w}_{(C_i, C_j)}$ is the average weight of edges connecting C_i and C_j . Due to formula (3), $\nabla_{den}(C_i) \leq 1$. The smaller the value of $\nabla_{den}(C_i) \leq 1$ is, the more different the topologies of C_i and the inter-part between C_i and C_j is. So does $\nabla_{den}(C_j)$.

Therefore, the similarity between two communities is computed as formula (4), which not only contains the connectivity degree between two communities but also considers the difference of local topology dynamically in clustering.

$$sim(C_i, C_j) = \frac{(\nabla_{den}(C_i) + \nabla_{den}(C_j))}{2} \times \overline{w}_{(C_i, C_j)} \quad (4)$$

3.4 CACNSC Algorithm

Finally, the paper proposes a robust agglomerative hierarchical algorithm, the Clustering Algorithm for Complex Network based on Semantics Core, named CACNSC. The algorithm first detects and expands semantics core. A semantics core is treated as a whole in the following clustering process, which reduces the complexity of hierarchical clustering. In the clustering, the most similar clusters based on connectivity and local topology are merged, until only K clusters remain. The overview of CACNSC is as follows:

Algorithm 2. CACNSC(G_{sem} , K)

- 1: Obtain the set C of semantics cores by performing Core_Detection(G_{sem});
 - 2: Treat each semantics core as a node and obtain the updated network $G_{sem-core}$;
 - 3: Obtain the set C^+ of expanded semantics cores according to Core_Detection($G_{sem-core}$);
 - 4: Treat each core of C^+ as a separate community and performing agglomerative hierarchical clustering following step 5-step 7;
 - 5: Compute similarities among communities according to formula (4);
 - 6: Merge the most similar clusters and compute the similarity between the new emerged cluster with the others;
 - 7: Repeat step 6 until only K clusters remain.
 - 8: The result clusters with the smallest size are treated as outliers.
-

Fig.2(d) shows the hierarchical clustering process of Fig.2.(a) with $K=2$. The computation complexity of CACNSC is analyzed as follows: the sorting complexity is $O(N \log N)$; it costs $O(N^2)$ in detecting semantics core; and the complexity of hierarchical clustering is $O(N^2)$ and the complexity of similarity computation of each merged community with others is $O((N - k) \times N)$. Thus the overall complexity is $O(N^2)$.

Compared with traditional clustering methods and clustering algorithms of complex network, CACNSC only need one parameter K and has relative low complexity. For instance, the complexity of traditional hierarchical clustering is $O(N^2)$. The complexity for FN[2] is $O((M + N)N)$, where M is the number of edges in the network and is usually much larger than N . The complexity of spectral clustering[13] is $O(KN^3)$.

4 Semantics Annotation Based on Clustering Tree

Semantics-based 3D model retrieval requires accurate semantics, which is usually represented by several keywords. Based on the semantics clustering tree of Section 3, this section aims at annotating automatically according to the semantics of limited 3D models. This section also states a method to recommend a few important 3D models for expert labeling, which forms a good foundation for automatic annotation.

The hierarchical clustering process of CACNSC forms a semantics tree T , which shows the semantics accumulation under different granularity. Fig.2.(d) is an example. Leaves of T correspond to the nodes that can be classified into 3 classes: (1) core node corresponding to semantics core, such as node $\text{Core}(v_1)$ and $\text{Core}(v_2)$ in Fig.2.(d). The weight of path that connects a core node with its child is equal to the weight of the corresponding edge in a complex network. For instance, the weight between tree node $\text{Core}(v_1)$ and v_3 is 3. (2) Expanded core node v_{core^+} , such as the father of $\text{Core}(v_2)$ in Fig.2.(d). The weight of path that connects a v_{core^+} with its child equals the weight of corresponding edge in complex network in $G_{\text{sem-core}}$. For example, the weight between $v_{\text{core}(v_1)^+}$ and v_9 is 1.33. (3) Non-leaf node is created by merging two communities in clustering. And the weight of the path that connects a non-leaf node with its child is equal to the similarity between its two children according to formula (4).

Therefore, semantics annotation of 3D models is equivalent to decide semantics of all leaf nodes in T . And the semantics of a model o_i is termed as $SM(o_i) = \{\text{keyword}_1^i, \dots, \text{keyword}_r^i\}$ with $r=1$ in the paper, which means using just one keyword as semantics. Since a community is a collection of models having similar semantics, we treat a community as the basic unit for annotation. After extracting a community C_i as a sub-tree T_i from T , the semantics annotation algorithm is stated as follows:

If a community has no semantics-known node, the semantics of its nodes is decided by its most similar community. Fig.3.(a) is an example of pruned tree of Fig.2.(d) along with path weight. Fig.3.(b) is the annotation result of Fig.3.(a), where only two nodes v_8 and v_9 have initial semantics "B" and "A" respectively. At the first step, v_9 's brother and its father are annotated with "A". Then, v_6 is annotated with "A", since the path weight between v_6 and v_9 's father is the highest. All nodes in $\text{Core}(v_1)$ are annotated with "B" according to the semantics of v_8 , although no node of $\text{Core}(v_1)$ is semantics known.

The complexity of semantics annotation is analyzed as follows: semantics clustering tree can be constructed during clustering and in the worst case a tree with $2N - 1$ nodes are obtained; it costs $O(2N - 1)$ to traverse T , therefore it costs $O((2N - 1)(N - N_{\text{known}}))$ at most, where N_{known} is the count of semantics-known nodes. Thus, its complexity is $O(N^2)$. It is much lower than an annotation method based on the shortest route of a graph, whose complexity is $O(N^3)$.

Algorithm 3. Semantic_Annotation(T_i)

- 1: Prune a node of (T_i) , if the semantics of all of its children are unknown;
 - 2: For the pruned T_i^- , first annotate the semantics of brothers of a semantic-known node. If more than one brother is semantics known, its semantics equals to that of the brother having the highest path weight in T_i^- ;
 - 3: Obtain the path the with highest weight between a semantics-unknown node and the known in T_i^- , the semantics of these two nodes are equal;
 - 4: Semantics of a pruned node equals to that of its father.
-

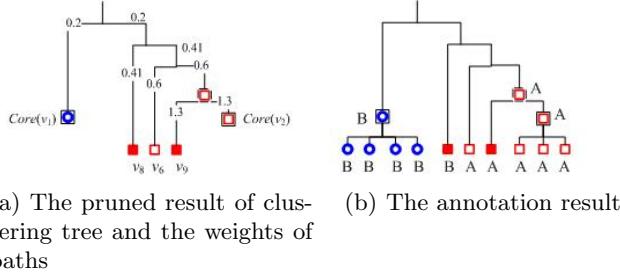


Fig. 3. The example of prune and annotation based on the semantic clustering tree

Automatic semantics annotation cannot be performed when no model has semantics. Therefore, this paper recommends a few important 3D models for expert labeling. The importance of a node can be decided by the ranking technology, such as PageRank [14]. Instead of applying PageRank to analyze the whole network, PageRank is used to recommend the most important node of each community as a representative. And K representatives are recommended. Since $K \ll N$, the labeling cost can be greatly reduced.

5 Experiment and Analysis

The experiment adopts a simulated dataset and the real feedback dataset of Princeton Shape Benchmark [4].

The simulated dataset forms a weighted complex network with quite different local topologies and lots of noises. The dataset contains 1000 feedbacks of 1000 objects, which belong to 6 classes with $n_1 = 400, n_2 = 250, n_3 = 150, n_4 = 100, n_5 = 50$ and $n_6 = 50$. Fig.4.(a) visualizes the relationship matrix M_{sem} .

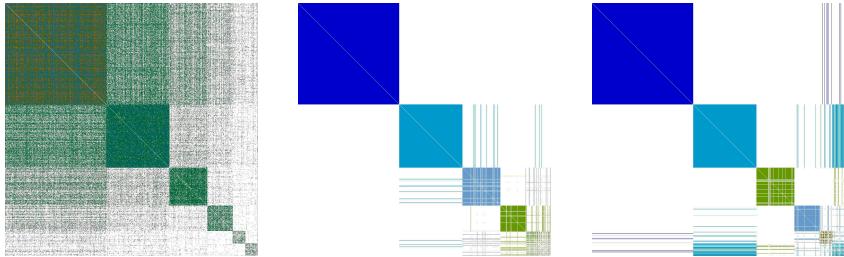
Three users (two men and one woman) provide 2721 feedbacks for 907 3D models ranging from m0 to m906 of PSB. The manual classification "Coarse-1" is treated as ground truth, which classifies these models into 60 semantics classes.

5.1 Clustering Performance

We use the criterions Entropy and Purity[15] to evaluate the clustering performance. The definition of entropy and purity is as follows:

$$Entropy = \sum_{i=1}^K \frac{n_i}{N} \left(-\frac{1}{\log l} \sum_{j=1}^l \frac{n_i^j}{n_i} \log \frac{n_i^j}{n_i} \right) \quad (5)$$

$$Purity = \sum_{i=1}^K \frac{1}{N} \max_j(n_i^j) \quad (6)$$



(a) visualization of semantic correlation matrix, the color of each pixel represents the value of sem
 (b) visualization of 6 clusters obtained by CACNSC
 (c) visualization of 6 clusters obtained by spectral clustering

Fig. 4. The visualization of M_{sem} of the simulated dataset and its clustering result

Where l is the count of the artificial classes and K is the number of result clusters, n_i^j is the count of the data which belongs to the j -th class and appears in the i -th cluster. The smaller the Entropy is and the bigger the Purity is, the better the clustering performance is.

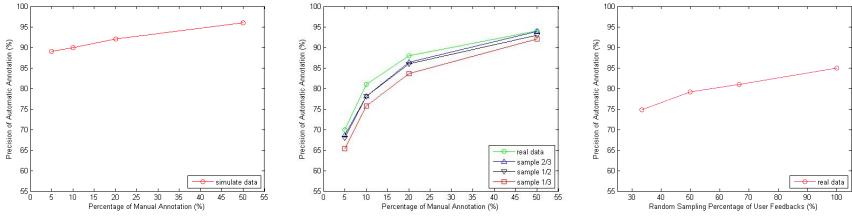
Table 1 shows the comparative result between CACNSC and other clustering algorithms. The spectral clustering method satisfying normalized cut [13], the FN method which is an expanding of modularity evaluation function in weighted network based on the idea of ref.[12] and X-means [17] are used to compare. Clustering result based on combined shape feature [16] that performs much better than single kind of feature is also stated. Fig.4.(b) and (c) visualizes the clustering results of CACNSC algorithm and spectral clustering algorithm. Obviously, CACNSC algorithm performs better for both real data and simulated data.

Table 1. Comparison of clustering results of different algorithms

dataset	Algorithm	Entropy	Purity	Remark
Simulated dataset	CACNSC	0.126	0.933	6 cluster,10 outliers
	Spectral clustering	0.155	0.927	6 cluster
	FN	0.466	0.660	2 cluster,18 outliers
Combined content-feature of PSB	X-means	0.217	0.216	62 cluster
User feedback of PSB	CACNSC	0.107	0.809	50 cluster,44 outliers
	Spectral clustering	0.142	0.766	50 cluster
	FN	0.012	0.006	4 cluster,96 outliers
	X-means	0.199	0.692	60 cluster

5.2 Semantics Annotation Performance

The automatic annotation performance is evaluated based on the clustering result of CACNSC.



(a) the annotating precision of the simulated dataset
 (b) the annotating precision of PSB
 (c) the annotation precision based on the semantics of recommended models

Fig. 5. The precision curve of automatic annotation

First, we test the annotation performance using different number of semantic-known objects. We randomly appoint 5%, 10%, 20% and 50% objects of simulated data and PSB as semantics-known models. For each ratio, the sampling process repeats 20 times to avoid performance fluctuation. Fig.5.(a) shows that the average precision is over 85% when only 5% objects have semantics for the simulated dataset.

Second, we test the annotation performance based on different clustering results. Fig. 5. (b) is the annotation evaluation of PSB, and the clustering results of CACNSC are obtained with $K=50$ based on the feedbacks sampled by ratio $\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\}$ from 2721 feedbacks. For each clustering result, annotation are performed according to the sampled semantics-known objects by ratio 5%, 10%, 20% and 50%. Even with only $\frac{1}{3}$ feedbacks, the precision achieves 65% with only 5% semantics-known models.

Third, the annotation performance based on the semantic of recommended objects is evaluated. The annotation precision reaches 80.2% for simulated dataset, when only 6 nodes are labeled. And Fig.5.(c) shows that the annotation precision of PSB achieves 75% based on the semantics of 50 recommended models while the feedbacks are sampled. We can see that the precision of automatic annotation is quite good, while the expert labeling is better.

6 Conclusion

The paper explores a complex-network-based strategy to extract semantics relationships, to analyze semantic clusters and to annotate precise semantics for 3D models. The semantics network comes from the implicit feedbacks. The proposed strategy performs quite well in semantics analysis and automatic semantics annotation. Our future work will concentrate on the evolution model of semantic complex network, which evolves with the increment of feedbacks.

Acknowledgments. This work is sponsored by the National Key Technology Research and Development Program of the Ministry of Science and Technology of China under grant number 2009BAH42B02, 2012BAH08B02, by the Natural Science Foundation of China under grant number 71272216, 60903080 and by the Fundamental Research Funds for the Central Universities under grant number HEUCF1212, HEUCF1208.

References

1. Atmosukarto, I., et al.: Feature Combination and Relevance Feedback for 3D Model Retrieval. In: Proceedings of the 11th International Multimedia Modeling Conference, pp. 334–339 (2005)
2. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. *Physical Review E* 69(6), 066133 (2004)
3. Jiang, W., Er, G., Dai, Q., Gu, J.: Hidden annotation for image retrieval with long-term relevance feedback learning. *Pattern Recognition (PR)* 38(11), 2007–2021 (2005)
4. Lv, T., et al.: Researches on Semantic Annotation and Retrieval of 3D Models Based on User Feedback. In: SKG 2010, pp. 211–218 (2010)
5. He, X., Jhala, P.: Regularized query classification using search click information, pp. 2283–2288. Elsevier Science Inc. (July 2008)
6. Lv, T., et al.: Analysis of community evaluation criterion and discovery algorithm of weighted complex network. *Acta Phys. Sin.* 61(21), 210511 (2012)
7. Clauset, A., Moore, C., Newman, M.E.J.: Hierarchical structure and the prediction of missing links in networks. *Nature* 453(7191), 98–101 (2008)
8. Shilane, P., et al.: The Princeton Shape Benchmark. In: The Shape Modeling International, pp. 388–399 (2004)
9. Barrat, A., et al.: Modeling the evolution of weighted networks. *Physical Review E* 70, 066149 (2004)
10. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *Proc. of the National Academy of Science* 9(12), 7821–7826 (2002)
11. Pan, X., Zhang, S., Ye, X.: A Survey of Content-Based 3D Model Retrieval with Semantic Features. *Chinese Journal of Computers* 32(6), 169–179 (2009)
12. Newman, M.E.J.: Analysis of weighted networks. *Phys. Rev. E* 70, 056131 (2004)
13. Shi, J.B., Malik, J.: Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 22(8), 888–905 (2000)
14. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the Web (January 29, 1998)
15. Zhao, Y., Karypis, G.: Criterion Functions for Document Clustering: Experiment and Analysis. Technical Report, University of Minnesota, pp. 01–40 (2001)
16. Bustos, B., et al.: Automatic Selection and Combination of Descriptors for Effective 3D Similarity Search. In: IEEE Sixth International Symposium on Multimedia Software Engineering, pp. 514–521 (2004)
17. Pelleg, D., Moore, A.: X-means: Extending K-means with efficient estimation of the number of clusters. In: Proc. 17th ICML, pp. 89–97. Stanford University (2000)

bCATE: A Balanced Contention-Aware Transaction Execution Model for Highly Concurrent OLTP Systems

Xiaogang Shi, Yanfei Lv, Yingxia Shao, and Bin Cui

Department of Computer Science & Key Lab of
High Confidence Software Technologies(Ministry of Education), Peking University
{sxg, lvyf, simon0227, cui.bin}@pku.edu.cn

Abstract. Web applications like social networking and online shopping are growing rapidly, forcing the OLTP systems to have the ability to efficiently handle numbers of concurrent transactions. Shared-everything models used in conventional OLTP systems, however, face significant problems in concurrency and scalability. With the increment of concurrent threads, the contention among threads increases sharply and degrades the system performance significantly. Shared-nothing models, on the other hand, are ideal for scalability but suffer a lot from data skew problem.

In this paper, we propose bCATE, a novel concurrent transaction execution model which divides the database into conflict partitions and detects the conflicts between transactions in each partition. bCATE adopts an efficient thread assignment strategy to alleviate the performance degradation caused by contention and data skew. We conduct extensive empirical studies on our implementation of bCATE on Shore-MT [1] and demonstrate that bCATE can achieve up to 50% performance promotion against other models.

Keywords: OLTP, high-concurrency, contention-aware, data skew, transaction execution.

1 Introduction

On-Line Transaction Processing (OLTP) Systems are widely used in a variety of web applications such as social networking and online shopping. As the number of users grows exponentially in recent years, OLTP systems are required to deal with a large number of concurrent requests. Therefore, the concurrent transaction management has become a critical issue for these applications.

Conventional OLTP systems, however, are suboptimal for this demand[1, 2]. In these systems, data is global and shared among all threads, as shown in Figure 1(a). User queries are directly dispatched to threads using simple rules like round robin, regardless of potential contention among executing queries. Locks and latches are adopted to ensure consistency. However, Pandis et. al. [3] figured out that the shared-everything transaction execution models used in conventional

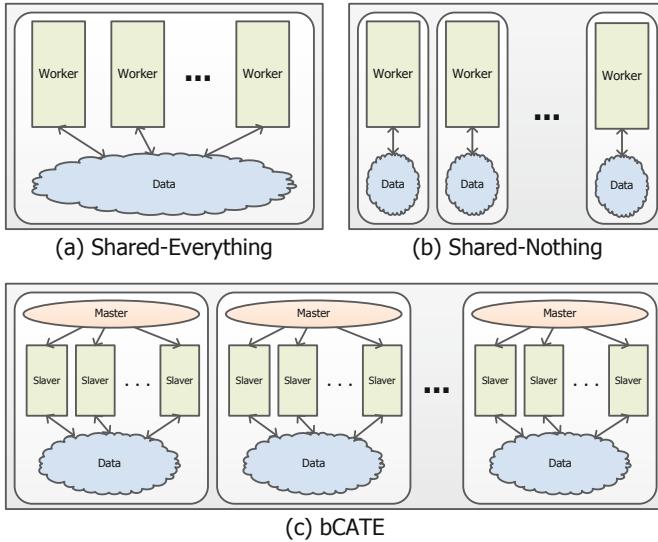


Fig. 1. shared-everything and shared-nothing transaction execution models and bCATE's design

OLTP systems limit the system's concurrency. Since the transactions executed by work threads are arbitrary and not coordinated, these worker threads contend with each other for shared resources. When the number of work threads increases, the degree of contention between threads increases at the same time, which downgrades the system's performance seriously.

In order to overcome the drawbacks of shared-everything models, Stonebraker et. al. proposed shared-nothing transaction execution models[4]. Different from shared-everything models, the data resources in the database are divided into partitions and each partition is allocated to one worker thread as shown in Figure 1(b). In this way, a partition can only be accessed by a single thread, and as a result, contentions among worker threads are significantly alleviated. Costly lock acquiring and releasing can be eliminated in shared-nothing models and overall performance is improved. Systems can also exploit extra parallelism by dividing a larger query into constituent smaller queries according to data partition and attached them to separate threads[5].

Although partitioning has many advantages, it also brings some problems. The performance of shared-nothing models is sensitive to data skew[6, 7]. A thread strictly corresponds to one partition in shared-nothing models, so in skewed workload where accesses concentrate on only a part of partitions, some threads are assigned excessive work while others are idle. This unbalanced assignment significantly deteriorates the overall performance and wastes the precious processor resource. Although some repartitioning strategies are proposed, these approaches incur heavy extra cost. Besides, many workloads can not be perfectly

partitioned. It's quite hard for each transaction to access only a single partition since the partition is relatively small. In this case, executing transactions need inter-partitions access, which requires costly distributed consensus protocols to ensure ACID property [8, 9].

In this paper, we propose bCATE (Balanced Contention-Aware Transaction Execution), a novel concurrent transaction execution model for highly concurrent OLTP systems. Our transaction execution model takes the advantages of both shared-everything and shared-nothing models by combining their merits. The architecture of bCATE is illustrated in Figure 1(c). bCATE divides the database into different conflict partitions and each conflict partition is assigned a master thread and a set of slave threads. Master threads are used to detect the potential contention among queries and slave threads are used to execute queries. Queries are firstly submitted to master threads to see if they conflict with any existing queries. If a query does not conflict with other queries, it then is delivered to a slave thread and executed by the slave thread.

Since all queries executed by slave threads don't conflict with any other executing queries, slave threads could execute without any blocks and waits. As a result, the processing capability wasted due to the contention among transactions is reduced in bCATE. Besides, as each conflict partition in bCATE is shared among a set of slave threads, bCATE is more resistant to skewed load compared with shared-nothing models. When the load is skewed across conflict partitions, bCATE can bypass costly repartitioning and just need to re-distribute slave threads among conflict partitions to balance the load. The contributions of this paper are:

1. We present a novel transaction execution model to reduce the total amount of wasted processing capability by partitioning and efficient conflict detection.
2. We take efficient approaches to get rid of the problems brought by data skew. We bypass the costly repartitioning to balance the system load by sharing each partition among a set of threads and re-distributing threads across partitions.
3. We implement our prototype in Shore-MT and evaluate the transaction execution model using various benchmarks. The experiments show that bCATE can attain higher throughput against conventional transaction execution models and it's more resistant to skewness.

The rest of this paper is structured as follows. In Section 2, we survey related work. We give an overview our transaction execution model in Section 3 and evaluate our model in Section 4. In Section 5, we conclude the paper.

2 Related Work

Many works have been done on transaction execution to improve the performance of OLTP systems. Shared-everything multi-threaded models are widely used to fully utilize CPU and I/O resources. As contention is the major performance bottleneck in shared-everything models, there is a significant amount of

effort toward reducing the contention in OLTP systems. Several previous studies focus on the contention in the data and instruction cache [10–12] while some other studies focus on the contention in the log manager. Johnson et al. proposed a novel log buffer [13] to improve log manager’s scalability. The contention in the lock manager is also studied. Johnson et al. [14] addressed that the lock manager is a main source of contention in OLTP systems and proposed a technique called SLI to bypass the bottleneck.

Partitioning is a primary way in scaling OLTP systems. A number of automatic partitioning schemes have been investigated in the past [15–17] and these schemes are orthogonal to our approach. Pandis et al. proposed a logical shared-nothing model called DORA [3] to eliminate locking in OLTP systems. Both DORA and bCATE use conflict detection to eliminate locking during execution, but DORA can only support simple partitioning schemes and suffers more from data skew than bCATE. In their later work, a physiological partitioning approach is proposed to eliminate page latching [18] which can also be applied in bCATE. Despite of the benefits, partitioning also brings a lot of problems including data skew and distributed transactions. Some researchers proposed complex concurrency control schemes [19, 20] to reduce the overhead caused by distributed transactions while other researchers attempted to minimize the number of distributed transactions by creating initial partitioning schemes [21, 9]. Repartitioning is a common way to alleviate data skew[22], but it incurs heavy extra cost due to data migration. By sharing a partition among a set of threads, bCATE is more resistant to skewness and could employ lightweight thread migration to balance the workload.

3 Design Overview

In the section, we will describe our transaction execution model in detail. To begin with, we give a brief view of our transaction execution model in Section 3.1. Detailed implementation in bCATE’s design is described in later sections and at last, we introduce the benefits of bCATE’s design in Section 3.4.

3.1 Transaction Execution

The execution of a transaction in bCATE is illustrated in Figure 2, in which the numbers indicate the processing steps in our model.

The database is first partitioned into different conflict partitions according to the partitioning schemes defined by users. bCATE provides a convenient interface for user to define their partitioning schemes. User queries, which are expressed in relational calculus, are first rewritten in the form of database operators and translated into a sequence of small and local queries on conflict partitions according to the partitioning schemes (step 1 in Figure 2). Then these local queries are routed to corresponding conflict partitions (step 2). Techniques for query decomposition and localization which are well studied in distributed databases can be applied here[23].

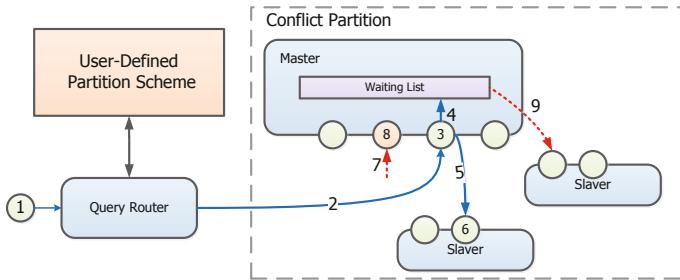


Fig. 2. Transaction execution in bCATE

Both master threads and slave threads act in an event-driven manner. When enqueued in a conflict partition, local queries are first delivered to the conflict partition's master thread. On receiving a local query, the master thread first detects the potential contention between this query and other executed queries to find if this query is executable (step 3). The detailed implementation of contention detection is introduced in section 3.2. A query isn't executable when it conflicts with any previously executed queries or currently executing queries. If a query isn't executable, it will be pushed into a waiting list (step 4); otherwise, it will be delivered to a slave thread (step 5).

Executable local queries are dispatched to slave threads as evenly as possible so that the load in the conflict partition can be balanced. Since there is no contention among executable local queries, slave threads can execute local queries in parallel without any waits or blocks (step 6).

When a transaction is aborted or committed, all the transaction's queries become committed and should be removed from all the corresponding partitions. The transaction sends messages to all the conflict partitions involving in its execution (step 7) when it terminates. When the master thread receives the message, it will commit all the transaction's queries in the conflict partition (step 8). After committing a executed query, some other queries conflicting with the query might become executable. The master thread probes the waiting list to find all those pending queries that can be executable now and dispatches these executable queries to slave threads (step 9).

3.2 Contention Detection

Contention is detected during query execution in conventional shared-everything models. When a worker thread in shared-everything models executes a query, it first picks up all the tuples the query will access at first and uses the identifiers of these tuples to acquire locks. The locks will be held by this work thread if there are no queries conflicting with the query.

This method doesn't work in bCATE. Different from shared-everything models, bCATE detects contention in master threads before queries are executed. Master threads have no knowledge of the tuples a query will access. A simple approach is to lock the whole partition, but it will significantly reduce the concurrency since a query will block all the queries with incompatible modes in the same partition even they are accessing different tuples.

We solve the problem by using predicate locking protocol[24]. Different from locking in conventional databases, predicates, but not tuples, are used as the units of locking. We illustrate how conflict detection works using a simple example. Supposing there are two queries:

Q1. *SELECT * FROM table WHERE id = 3*

Q2. *DELETE * FROM table WHERE id > 2*

Q1's predicate is $id = 3$ and Q2's predicate is $id > 2$. As the tuples satisfying Q1's predicate also satisfy Q2's predicate, it's implied that Q1 would access the tuples which may be accessed by Q2 simultaneously. Since Q1's lock mode (Share) and Q2's lock mode (Exclusive) are not compatible, it will lead to potential contention if Q1 and Q2 are executed concurrently. Therefore, we can say that Q1 conflicts with Q2 and if Q2 is executed but not uncommitted in the system, Q1 would be put in the wait list and can't be executed until Q2 commits.

bCATE keeps a list of all the executed but uncommitted queries in each partition. Whenever a new query arrives, the master thread checks the query's compatibility with all the executed queries in the list using predicate locking protocol. If the query doesn't conflict with all the queries in the list, then the query is executable and will be dispatched to a slave thread. Since the number of uncommitted queries is limited in the partition, the overhead of conflict detection is relatively low compared to the overhead of query execution.

3.3 Load Balancing

Though bCATE is more resistant to skew than shared-nothing models, bCATE still needs to balance the load in case that the performance is hurt by data skew. Shared-nothing systems usually balance the load by costly repartitioning the database according to the load distribution. bCATE's design allows us to have a lightweight but efficient load balancing mechanism.

bCATE monitors the partition's load and calculates the ideal number of each partition's slave threads. Let P be the total number of partitions, L_i be the load of the i th partition and S_i be the number of slave threads in i th partition. In an ideal situation, the number of each partition's slave threads should be adaptive to the partition's load. So the ideal number of slave threads for each partition by:

$$I_i = \frac{\sum_{i=1}^n S_i}{\sum_{i=1}^n L_i} \times L_i.$$

When the number of the partition's slave threads S_i is less than $I_i - \Delta$ for a given upper bound Δ , we increase the number of this partition's slave threads

to the ideal number I_i . When the number of the partition's slave threads S_i is greater than $I_i + \nabla$ for a given lower bound ∇ , we reduce the number of this partition's slave threads to the ideal number I_i .

As each partition is assigned with a reasonable number of slave threads, the load in the partition can be distributed among these threads, which alleviates the performance degradation due to data skew.

3.4 Benefits of bCATE

Since data may be accessed by different threads simultaneously in shared-everything models, the worker threads must first probe the lock table and acquire locks before executing to ensure correctness. The lock can't be acquired at once if the query conflicts with any executing queries and the worker thread will have to wait until the lock is acquired (shown in Figure 3(a)). As a result, the processing capacity is wasted due to unnecessary waiting.

Different from shared-everything OLTP models, data can only be accessed by a single thread in shared-nothing models, so worker threads don't need to acquire locks for executing queries. However, the parallel capacity inside a partition is scarified since the queries in the same partition have to be executed by a thread serially (shown in Figure 3(b)). It gets worse when the load is skewed in the system. Threads in heavy-loaded partitions are busy executing while threads in light-loaded partitions are idle.

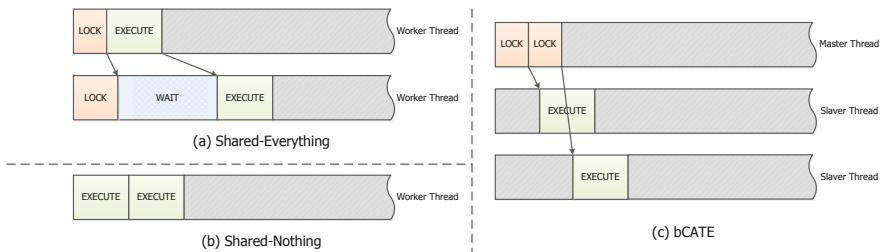


Fig. 3. Query processing in shared-everything, shared-nothing and bCATE

bCATE outperforms both of them in terms of the efficiency of query processing. By separating query execution into two phases, threads in bCATE can reduce the performance degradation due to unnecessary waits. When a query is not executable, master threads in bCATE will not wait and just move forward to process the next queries in the queue. As the queries don't conflict with each other, slave threads can execute these queries in parallel perfectly, fully utilizing the computing capacity. Furthermore, by allowing a set of threads to execute queries in parallel, bCATE can exploit more parallelism in query processing. Though contention detection can not be avoided in bCATE and contention is detected in master threads serially, bCATE still benefits more from executing

queries in parallel because the overhead of contention detection is relatively low compared to query execution.

Though bCATE still suffers the problems brought by partitioning, it's less painful for bCATE than for shared-nothing models. The increased load in the affected partitions can be distributed among slave threads. Besides, as there is no resources bound to slave threads in bCATE, it's easier to move slave threads than data resources from a partition to another partition. By redistributing slave threads among partitions, it makes bCATE more efficient to balance the load than shared-nothing systems.

4 Evaluation

In order to evaluate our transaction execution model, we implement a prototype bCATE system on top of Shore-MT[1]. Since Shore-MT does not have a SQL front end, all the transactions are hard-coded. We also implement a shared-everything (SE) model and a state-of-the-art shared-nothing (SN) model [3] on Shore-MT as the competitors in our experiments. To reduce the influence brought by partitioning schemes, we use horizontal partitioning schemes for both the shared-nothing model and bCATE. The database is partitioned into 24 partitions in the shared-nothing model to fully utilize the CPU cores while the database is partitioned into 4 partitions and each partition is assigned 8 slave threads in bCATE.

4.1 Experiment Setup

We perform all our experiments on a Lenovo R680 server running CentOS 5.4 with 48GB of RAM and two processors of Intel E7-4807 processors, each of which contains six cores clocked at 1.86GHz. We configure Shore-MT with 4GB buffer to ensure that the database size fits in the buffer pool size. As such, the flushing of log is the only I/O in the system.

Two benchmarks are adopted in our evaluation: Nokia's Network Database Benchmark [25] (TATP, also known as "TM1") and Sysbench [26].

TATP is an open source workload designed specifically for high-throughput applications. TATP contains eight kinds of extremely short transactions. 80 percent of the transactions are read-only transactions and the remaining 20 percent are update transactions. There's almost no range query in TATP's transactions. We use a 1,000,000 subscriber TATP data set occupying approximate 1GB in our experiments.

Sysbench is a multi-threaded benchmark tool for evaluating database server performance. We use a Sysbench data set with 1,000,000 rows. There are five types of transaction in our experiments including Point Queries, Range Queries, Update Queries, Delete Queries and Insert Queries. The portion of Point Queries is 80 percent and the others are 5 percent, respectively.

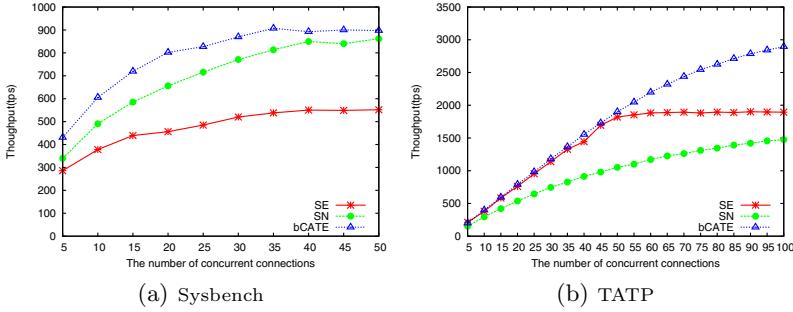


Fig. 4. Performance of variable models as the number of concurrent threads increases

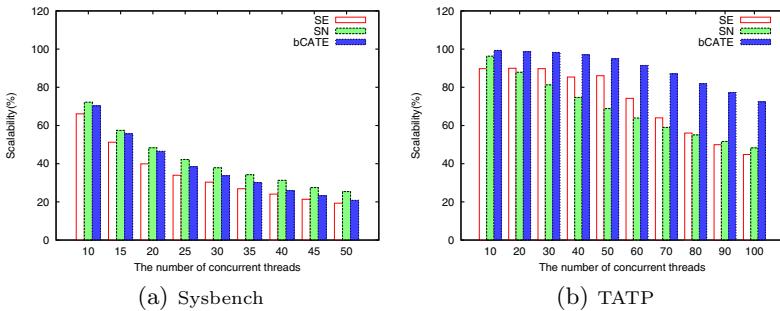


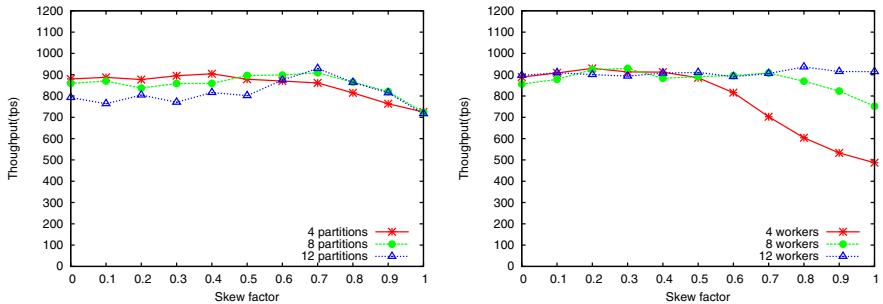
Fig. 5. Scalability of different models as the number of concurrent threads increases

4.2 Concurrency and Scalability

We first quantify how effectively bCATE improves the concurrency and scalability for highly-concurrent OLTP systems by comparing the shared-everything model (SE), the shared-nothing model (SN) and our bCATE model. The number of transactions executed is employed as the criterion. Then scalability on each model is calculated by dividing its throughput by the performance in the system when there is a single thread. The performance of three models as the number of concurrent threads in system increases is shown in Figure 4 and Figure 5 shows the scalability of these models.

In general, bCATE exhibits good performance in the presence of highly concurrent workloads as bCATE can eliminate the potential contention and exploit extra parallelism when executing transactions. It can be seen that the shared-everything model experiences serious concurrency and scalability problems with the increment of concurrent threads in Sysbench. The shared-nothing model exhibits a better scalability in Sysbench while bCATE outperforms the shared-nothing model by exploiting extra parallelism in partitions.

When evaluating these models in TATP, bCATE stands out in both performance and scalability. Transactions in TATP are extremely short and they merely conflicts with each other. As a result, the shared-nothing model suffers



(a) Different number of partitions in the system (b) Different number of workers in a partition

Fig. 6. Performance of variable bCATE's configurations as the skewness of load increases in Sysbench

more from the extra cost brought by partitioning than the benefits from contention elimination. Both shared-everything model and bCATE scale well when the number of concurrent threads is small. But when the number of concurrent threads continues to increase, the shared-everything model's throughput reaches its peak while bCATE's throughput remains increasing.

4.3 Tolerance to Skew

Data skew is common in real workloads. To evaluate the three models' performance in skewed workload, we use Zipfian distribution with different skew factor to generate imbalanced accesses.

We evaluate bCATE's performance in skewed workload with various configurations. First, we partition the database into different numbers of partitions with the same number of slave threads. The performance of these configurations as skew factor increases is illustrated in Figure 6(a). As the skew factor increases, these configurations exhibit similar performance. Therefore, bCATE's performance in skewed load is not as sensitive as shared-nothing models to partitioning schemes.

Next, we partition the database into the same number of partitions and assign different number of slave threads to these partitions. Since each partition in bCATE is shared among a set of slave threads, the skewed load inside a partition can be naturally distributed across these threads. Therefore, the configurations with more slave threads are expected to tolerate more data skew because there are more slave threads in each partition. Figure 6(b) confirms our expectations. We can observe that the configuration with more threads in a partition tolerates more skewness. Thus, increasing the number of slave threads in heavy-loaded partitions can efficiently eliminate the performance degradation due to data skew.

Finally, we examine the impact of data skew on all the three models as illustrated in Figure 7. With the increment of skewness, the performance of shared-nothing model downgrades sharply, whereas the shared-everything model and

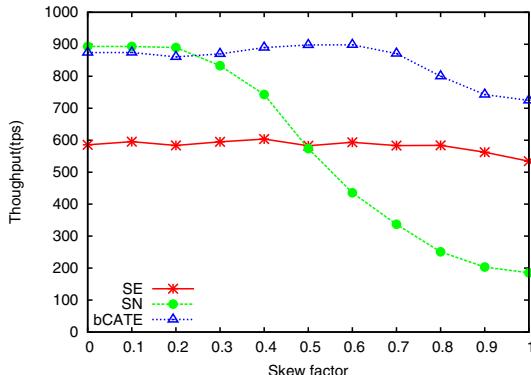


Fig. 7. Performance of different models as the skewness of load increases in Sysbench

bCATE provide stable performance. When the skew factor exceeds 0.7, most queries are routed to a single partition. In this case, the throughput of bCATE drops as the slave threads in the partition cannot complete the queries in time.

5 Conclusion

In this paper, we proposed a novel transaction execution model called bCATE to improve the concurrency and scalability for highly concurrent OLTP systems. Based on the observation that conventional shared-everything models suffer a significant performance degradation due to contention between threads, we eliminated the overhead by partitioning and contention detection. Sharing in the low level makes bCATE naturally resistant to data skew. The evaluation shows that a significant amount of throughput improvement is obtained for high-concurrent workloads and the performance is less influenced by data skew in bCATE.

Acknowledgements. This research was supported by the National Natural Science Foundation of China under Grant No. 61073019 and 61272155.

References

1. Johnson, R., Pandis, I., Hardavellas, N., Ailamaki, A., Falsafi, B.: Shore-mt: A scalable storage manager for the multicore era. In: EDBT, pp. 24–35 (2009)
2. Harizopoulos, S., Abadi, D.J., Madden, S., Stonebraker, M.: Oltp through the looking glass, and what we found there. In: SIGMOD, pp. 981–992 (2008)
3. Pandis, I., Johnson, R., Hardavellas, N., Ailamaki, A.: Data-oriented transaction execution. Proceedings of the VLDB Endowment 3(1-2), 928–939 (2010)
4. Stonebraker, M., Hachem, N., Helland, P.: The end of an architecture era (it's time for a complete rewrite). In: VLDB, pp. 1150–1160 (2007)
5. Cieslewicz, J., Ross, K.A.: Data partitioning on chip multiprocessors. In: DaMon, pp. 25–34 (2008)

6. Hua, K.A., Lee, C.: Handling data skew in multiprocessor database computers using partition tuning. In: VLDB, pp. 525–535 (1991)
7. Tözün, P., Pandis, I., Johnson, R., Ailamaki, A.: Scalable and dynamically balanced shared-everything oltp with physiological partitioning. *The International Journal on Very Large Data Bases*, 1–25 (2012)
8. Helland, P.: Life beyond distributed transactions: an apostate’s opinion. In: CIDR (2007)
9. Pavlo, A., Curino, C., Zdonik, S.: Skew-aware automatic database partitioning in shared-nothing paralleled oltp systems. In: SIGMOD, pp. 61–72 (2012)
10. Ailamaki, A., DeWitt, D.J., Hill, M.D., Wood, D.A.: Dbmss on a modern processor: Where does time go? In: VLDB, pp. 266–277 (1999)
11. Harizopoulos, S., Ailamaki, A.: A case for staged database systems. In: CIDR (2003)
12. Hardavellas, N., Pandis, I., Johnson, R., Mancheril, N., Ailamaki, A., Falsafi, B.: Database servers on chip multiprocessors: Limitations and opportunities. In: CIDR, pp. 79–87 (2007)
13. Johnson, R., Pandis, I., Stoica, R., Athanassoulis, M., Ailamaki, A.: Scalability of write-ahead logging on multicore and multisocket hardware. *The VLDB Journal* 21(2), 239–263
14. Johnson, R., Pandis, I., Ailamaki, A.: Improving oltp scalability with speculative lock inheritance. *Proceedings of the VLDB Endowment* 2(1), 479–489 (2009)
15. Zilio, D.C.: Physical database design decision algorithms and concurrent reorganization for parallel database systems. PhD thesis, University of Toronto (1998)
16. Rao, J., Zhang, C., Megiddo, N., Lohman, G.: Automating physical database design in a parallel database. In: SIGMOD, pp. 558–569 (2002)
17. Agrawal, S., Narasayya, V., Yang, B.: Integrating vertical and horizontal partitioning into automated physical database design. In: SIGMOD, pp. 359–370 (2004)
18. Pandis, I., Tözün, P., Johnson, R., Ailamaki, A.: Plp: Page latch-free shared-everything oltp. *Proceedings of the VLDB Endowment* 4(10), 610–621 (2011)
19. Jones, E.P., Abadi, D.J., Madden, S.: Low overhead concurrency control for partitioned main memory databases. In: SIGMOD, pp. 603–614 (2010)
20. Pavlo, A., Jones, E.P.C., Zdonik, S.: On predictive modeling for optimizing transaction execution in parallel oltp systems. *Proceedings of the VLDB Endowment* 5(2), 85–96
21. Curino, C., Jones, E., Zhang, Y., Madden, S.: Schism: a workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment* 3(1-2), 48–57 (2010)
22. Sockut, G.H., Iyer, B.R.: Online reorganization of databases. *ACM Computer Survey* 41(3), 14:1–14:36
23. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 3 edn. Springer (2011)
24. Eswaran, K.P., Gray, J.N., Lorie, R.A., Traiger, I.L.: The notions of consistency and predicate locks in a database system. *Communications of the ACM* 19(11), 624–633
25. TATP: Telecom application transaction processing benchmark,
<http://tatpbenchmark.sourceforge.net/>
26. SysBench: System performance benchmark, <http://sysbench.sourceforge.net/>

Personalized News Recommendation Using Ontologies Harvested from the Web^{*}

Junyang Rao, Aixia Jia, Yansong Feng^{**}, and Dongyan Zhao

ICST, Peking University, China

{raojunyang, jiaaixia, fengyansong, zhaodongyan}@pku.edu.cn

Abstract. In this paper, we concentrate on exploiting background knowledge to boost personalized news recommendation by capturing underlying semantic relatedness without expensive human involvement. We propose an *Ontology Based Similarity Model* (OBSM) to calculate the news-user similarity through collaboratively built ontological structures and compare our approach with other ontology-based baselines on both English and Chinese data sets. Our experimental results show that OBSM outperforms other baselines by a large margin.

Keywords: Recommender system, Content-based filtering, User profiling.

1 Introduction

The vast amount of news data available on the Internet has urged the development of personalized news recommender systems which are capable of providing individual users with personalized articles according to their interests. As one of the two most popular recommendation paradigms, *content-based filtering* has been widely used in news recommendation[1]. The process basically consists of comparing the user profile against the content of candidate articles and ranking them consequently according to users' reading interests.

In most content-based systems, all words in an article, except stop words, are represented in a *bag-of-words* vector and further embedded in a cosine similarity function to estimate the relationship between articles and user profiles. As a result, two news articles with few common words would be judged dissimilar in such a style, although they may be closely related to each other. These motivate the exploitation of semantic analysis in content-based systems.

Semantic analysis allows learning the similarities between articles and user profiles beyond traditional *bag-of-words* format, e.g., modeling the relatedness between two concepts according to real world knowledge. For instance, a basketball fan definitely knows *Michael Jordan*, a former NBA player, is regarded as one of the greatest players in the history of *Chicago Bulls*. But without these background in *basketball* and *NBA*, a system would never think *Michael Jordan* and current *Chicago Bulls* stars are related, and therefore it would never recommend a *Michael Jordan* article to a basketball fan

* This work is partially supported by the 863 Program (No. 2012AA011101) and the Natural Science Foundation of China (No. 61272344 and 61202233).

** Corresponding author.

with *Chicago Bulls* and *Derrick Rose*(a current Chicago star) in his/her user profile. The background knowledge can be in the form of finely crafted knowledge bases for specific domains, but are costly and time-consuming to build. Or, it may be coarse grained ontological structures that may contain noises but can be obtained from the Web without much human involvement. We argue that it is crucial to introduce real world knowledge into the content-based paradigms, but it is not expected to rely on expensively built knowledge bases. We, therefore, should first find out what background knowledge resources we can use without relying on intensive annotations, and secondly, how we can properly incorporate those background knowledge into a content-based framework, and lastly, how about the efficiency of adding semantic analysis.

2 Related Work

In content-based recommender systems, finely crafted ontologies or taxonomies have been attempted to model user profiles and estimate the similarity between two concepts.

Goossen et al. propose an extension of the classic TF-IDF approach, called CF-IDF[2]. The system works on a manually built NASDAQ ontology, considers its concept words to represent the article, and uses cosine similarity to model the relationship between user profiles and articles. *News@hand* [3] constructs a similar framework, but uses the Jaccard similarity. IJntema et al. define the similarity between two given concepts as the *cosine similarity* of their corresponding *semantic neighbourhoods*[4]. Degemmis et al. represent a book or article as *bag-of-synsets* according to WordNet, and further use a Naive Bayesian binary classifier to judge an item as interesting or not for a given user[5].

The methods discussed above represent articles and profiles as bag of concepts according to their ontologies, and simply operate the news-user matching as cosine or Jaccard similarities without any considerations about their embedded ontological structures. Furthermore, these ontologies are usually built manually for specific domains, thus limited in size and coverage. Therefore even harder to make a conclusion whether these approaches are easy to adapt to other domains or applications.

In this paper, we will model articles and user profiles on larger real world ontologies harvested from the web without much human annotations, and further investigate the concept similarity and news-user matching over such an ontology by considering its naturally embedded ontological structures.

3 Ontologies Built from Online Encyclopedia

Recently, there are many open domain knowledge resources available on the Web. Examples of general purposes include Wikipedia, DBpedia, Freebase, etc., in English, and Hudong encyclopedia, Baidu encyclopedia, etc., in Chinese. DBpedia is constructed from Wikipedia with more than 3.64 million things, 1.83 million of which are classified in a consistent ontology. Baidu encyclopedia and Hudong encyclopedia are similar to Wikipedia, both of which claim to host over 5 million concepts in Chinese. We thus build two world ontologies from those resources, one in English derived from DBpedia and the other in Chinese from Hudong encyclopedia.

The DBpedia release contains two files, one storing all classes in a hierarchy, and the other with all instances in the form of *subject-relation-object*. We build the DBpedia based ontology as: a), construct the inner nodes of the ontology from the hierarchy of the first file, and b), append all instances in the second file as leaf nodes of the ontology (see Figure 1 for illustration).

In Hudong encyclopedia, each concept appears as a web page containing its relationship with other concepts; concepts of the same category are stored in the same directory, and all directories form a hierarchy. We construct the Hudong based ontology as: a) build the skeleton of the ontology by scanning the hierarchy; b) extract instances and relations from every concept page and append them as leaf nodes of the ontology. Compared to the English one, we admit our Chinese ontology is noisy in nature, since it contains more duplicate concepts with different paths.

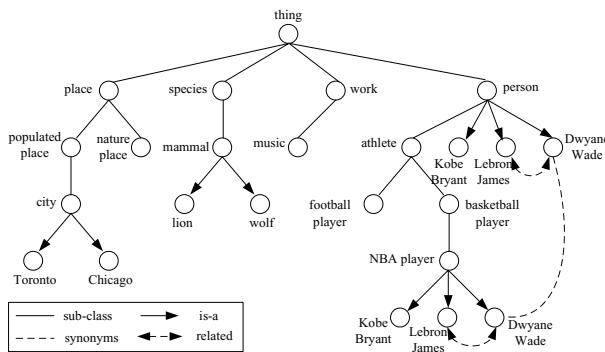


Fig. 1. A snapshot of the ontology based on DBpedia: we use *related* to represent the actual relation names for brevity

4 Ontology Based Similarity Model

In this section, we describe how we take advantage of background knowledge when measuring the similarities between news articles and user profiles. Formally, we assume the ontology built in section 3 contains a set of n concepts with their relations denoted as: $Ontology^W = \{c_1^w, c_2^w, \dots, c_n^w\}$. For a given article, we only consider its concept words appearing in the ontology concept list, represented as: $News = \{< c_1^n, w_1^n >, \dots, < c_p^n, w_p^n >\}$, where $c_i^n \in Ontology^W$, w_i^n is the TF-IDF weighting of concept c_i^n ($1 \leq i \leq p$), and p the number of concepts in the article. For a given user, we construct the user profile by accumulating all concepts found in the articles that the user has read before, similarly denoted as: $User = \{< c_1^u, w_1^u >, \dots, < c_q^u, w_q^u >\}$, where q is the total number concepts in the user's reading history, w_j^u is the average weighting of concept c_j^u in the articles containing this concept and read by this user.

Concept-Concept Similarity. Most existing ontology-based methods compute the similarity between two concepts sets using either cosine or Jaccard similarity functions, where concepts are treated isolated and measured according to their presence, i.e., a

concept will contribute 1 if occurring in both sets, otherwise 0, without any considerations about the semantic relationship between two concepts.

In this paper, the news-user similarity boils down to investigating the underlying semantic relatedness between two concepts. We propose to model the similarity between two concepts by taking their background ontological structures into account:

$$Csim(c_1, c_2) = \begin{cases} 1, & d = 0 \text{ or } isSynonyms(c_1, c_2) \\ \frac{e^\delta}{e^\delta + 1} \cdot (-\log_{2H} \frac{d}{2H}), & \text{otherwise} \end{cases} \quad (1)$$

where, c_1 and c_2 are two concepts, d is the shortest distance from c_1 to c_2 on the world ontological structure, while δ is the shortest distance from their lowest common ancestor to the root node of the ontology, H the height of the ontology.

In Formula 1, $-\log_{2H} \frac{d}{2H}$ has negative correlation with d and prefers two adjacent concepts. $\frac{e^\delta}{e^\delta + 1}$ is considered as a weight of $-\log_{2H} \frac{d}{2H}$ and will prefer two more concrete concepts at the bottom of the ontology, based on the assumption that two adjacent concrete concepts at a lower level tend to be more similar than at a higher level, since they share more common information from their ancestors.

News-User Similarity. Now we are ready to compute the similarity between a news article and a user profile as:

$$sim(\text{News}, \text{User}) = \frac{1}{p} \sum_{i=1}^p \max_{1 \leq j \leq q} \{ Csim(c_i^n, c_j^u) \times w_{i,j} \} \quad (2)$$

where $w_{i,j}$ is taken as a confidence of $Csim(c_i^n, c_j^u)$,

$$w_{i,j} = \frac{2}{1 + e^{k\tau}}, \text{ with } \tau = \frac{\text{abs}(w_i^n - w_j^u)}{\max(w_i^n, w_j^u)}. \quad (3)$$

We can see that when w_i^n and w_j^u are about equal, c_i^n and c_j^u have similar importance to their corresponding concept sets, the concept similarity $Csim(c_i^n, c_j^u)$ will have a higher confidence ($w_{i,j}$). The smoothing factor k in Formula 3 is used to control the sensitivity of the confidence factor $w_{i,j}$. Larger k leads to a sensitive confidence function, which in turn penalize the concept pairs that have distinct importance in their own sets.

Now with Formula 2 at hand, we are ready to compute the similarities between news articles and user profiles, and the articles with similarities over a threshold will be recommended to the users.

5 Evaluation

In this section, we evaluate our approaches and several traditional ontology-based baseline models on both English and Chinese datasets on a news recommendation platform (NRS) we developed. As of today, our NRS has about 581 users.

English Dataset. We construct the English ontology from DBpedia with 3.6 million things, and randomly select 6,000 articles from New York Times (2006-2007) for our English experiments. **Chinese Dataset** We construct our Chinese ontology with 5 million entries extracted from Hudong encyclopedia and around 6,000 articles are collected **everyday** from Sina News for our Chinese experiments.

Experimental Setup. We use CF-IDF[2] and Jaccard[4] as our baselines. Both methods model articles and user profiles in a *bag-of-concepts* format, and calculate news-user similarity by either cosine or Jaccard similarity. We perform our English experiment and Chinese experiment, respectively, as follows: 1) 400 news articles are randomly selected by NRS **at one time**, and shown to the users. Every user has to read all 400 articles and indicates whether it is interesting or not for each article. 2) Then, the 400 articles with ratings (interesting or uninteresting) are randomly split into two sets. 60% for training, 40% for testing. There are relatively equal proportions of interesting articles in each set. 3) For each user, NRS constructs the profile from the interesting articles of the training set. The test set is used by different algorithms to calculate the news-user similarity. An article is marked as interesting if its similarity is higher than a threshold. 4) For each article in the test set, we compare the model generated ratings with the human ratings, and compute Precision, Recall and F-measure for all methods. 5) In order to obtain reliable results, we ask NRS to repeat the evaluation process for 2000 iterations by randomly splitting the training/testing sets in each iteration, and calculates the average performance for each algorithm.

5.1 Precision, Recall and F-Measure

After the experiments are conducted, the precision, recall and F-measure are calculated by NRS. Table 1 shows that in the experiment over English and Chinese data sets, OBSM outperforms other algorithms on all aspects by a large margin. This is not surprising: our model utilizes the structured information embedded in the ontologies, and provides more reasonable similarity estimations between concepts, which is obviously better than using these concepts in a *bag-of-words* format.

Table 1. The Performances of different models in the English and Chinese experiments

Dataset	DBpedia & New York Times			Hudong & Sina News		
	Precision	Recall	F-measure	Precision	Recall	F-measure
CF-IDF	63.94%	77.81%	70.20%	63.10%	76.50%	69.16%
Jaccard	69.1%	76.56%	72.64%	71.15%	72.05%	71.60%
OBSM	74.5%	90.56%	81.75%	72.87%	81.71%	77.04%

In order to investigate the overall performance of these algorithms over different thresholds, we plot the the curve of the F-measure against different thresholds from 0.00 to 1.00. As figure 2(a) and 2(b) show, OBSM scores higher than others consistently. The standard threshold used for testing is set to 0.5, where OBSM has the highest F-score.

5.2 Precision-Recall Curves

When PR curves for these algorithms are plotted in one figure, we can see the difference clearly. As figure 2(c) and 2(d) show, in the experiment over both English data set and Chinese data set, OBSM performs better than CF-IDF and Jaccard.

OBSM performs a bit lower on the Chinese dataset than the English dataset, but others perform almost the same on both datasets. The reason may be that the ontology of Hudong encyclopedia is coarsely crafted as we mentioned in section 3, while DBpedia takes a more clear and less duplicate ontology. Recall that CF-IDF and Jaccard just use bag of concepts in the ontology and do not take any structure information of the ontology into account, while OBSM calculates the similarity between the news article and user profile by utilizing the structure of the ontologies. That is why CF-IDF and Jaccard show consistent performances no matter how messy the ontological structure is, while OBSM performs worse on the Chinese dataset than the English one.

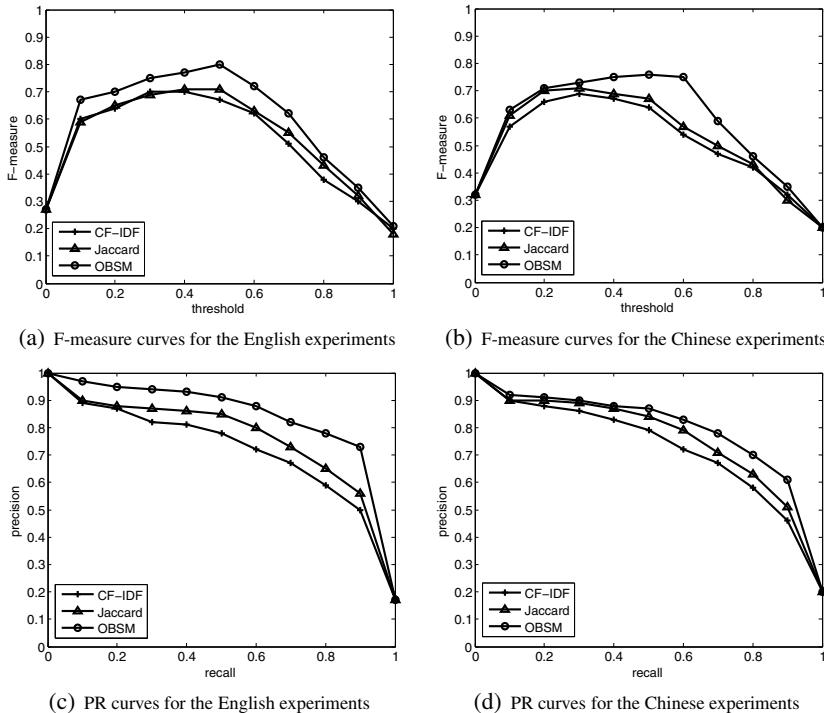


Fig. 2. The comparison of F-measures and PR curves over both English and Chinese datasets

6 Conclusion

In this paper, we exploit semantic analysis in the content-based filtering framework to boost personalized news recommendation by employing background knowledge in the

form of ontological structures automatically obtained from the web without expensive human annotations. This is novel to the best of our knowledge to make use such kind of resources in RS. We compared our model against several traditional ontology-based approaches in a news recommender system (NRS) over both English and Chinese datasets, and the results show that OBSM outperforms the baseline models on all measures over both datasets.

References

1. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.): *Recommender Systems Handbook*. Springer (2011)
2. Goossen, F., IJntema, W., Frasincar, F., Hogenboom, F., Kaymak, U.: News personalization using the cf-idf semantic recommender. In: WIMS, p. 10 (2011)
3. Cantador, I., Bellogín, A., Castells, P.: News@hand: A semantic web approach to recommending news. In: Nejdl, W., Kay, J., Pu, P., Herder, E. (eds.) AH 2008. LNCS, vol. 5149, pp. 279–283. Springer, Heidelberg (2008)
4. IJntema, W., Goossen, F., Frasincar, F., Hogenboom, F.: Ontology-based news recommendation. In: EDBT/ICDT Workshops (2010)
5. Degemmis, M., Lops, P., Semeraro, G.: A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. *User Model. User-Adapt. Interact.* 17(3), 217–255 (2007)

TinyQP: A Query Processing System in Wireless Sensor Networks

Shangfeng Mo^{1,2,3}, Hong Chen^{1,2,*}, Xiaoying Zhang^{1,2}, and Cuiping Li^{1,2}

¹ Key Laboratory of Data Engineering and Knowledge Engineering of MOE, Renmin University of China, Beijing 100872, China

² School of Information, Renmin University of China, Beijing 100872, China

³ Hunan University of Science and Technology, Xiangtan 411201, China

moshangfengxy@yahoo.com.cn,

{chong, zhangxiaoying2011}@ruc.edu.cn, cuiping_li@263.net

Abstract. Wireless Sensor Networks (WSNs) can be viewed as a new type of distributed databases. Data management technology is one of the core technologies of WSNs. In this demo we show a Query Processing system based on TinyOS in WSNs, which is called TinyQP. TinyQP provides the in-network processing of several complex queries including real time query, historical query, event detection, aggregate query, top-k query, etc. TinyQP displays the network topology in a graphical way. TinyQP allows the user to remotely execute query commands and browse query result.

Keywords: WSNs, Query Processing, complex query algorithms.

1 Introduction

Wireless Sensor Networks (WSNs) have broad applications in the fields of disaster warning, environment monitoring, etc. WSNs can be viewed as a new type of distributed databases. Data management is one of the core technologies of WSNs.

Nowadays there are some prototype systems. TinyDB [1] and Cougar [2] implement several basic query functions, such as aggregate (MAX, MIN, AVG, etc.), basic SQL-like query processing, etc. TinyCasper [3] monitors moving objects in WSNs while preserves their location privacy. KSpot [4] implements both snapshot and historical top-k queries in WSNs. These prototype systems implement several simple query functions or only support a separate complex query algorithm. We implements a Query Processing system (TinyQP) based on TinyOS [5] in WSNs. TinyQP simultaneously supports several complex query algorithms, supports network topology display and allows user to access the data of multiple WSNs through internet.

2 TinyQP Design and Implementation

2.1 TinyQP Architecture

Figure 1 depicts the system architecture of TinyQP. The TinyQP system includes three types of servers: Web server, application server and database server.

* Corresponding author.

The user sends query request to the Web server through the internet. The Web server forwards the request to the application server. The application server analyses the query request using the query parser. If the query request need to access the data of other sink, the application server send the query request to the application server of other sink. If the query request need to access local data, the application server produces an optimized query scheme. Then the scheme is saved in the database server. The application server communicates with sink node through the serial port. The application server sends query command to the sink node. The sink node broadcasts the query command to all nodes in the WSNs. The ordinary node senses the attribute values and transmits them to the sink node according to the query command. The sink node returns result to the application server. The application server processes the data and saves the data to the database server, and then returns the required data to the Web server. The Web server returns the required data to the user through internet.

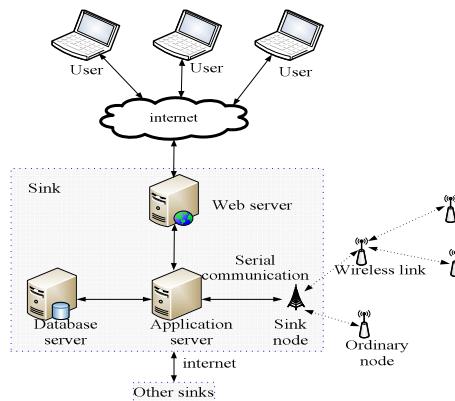


Fig. 1. TinyQP architecture

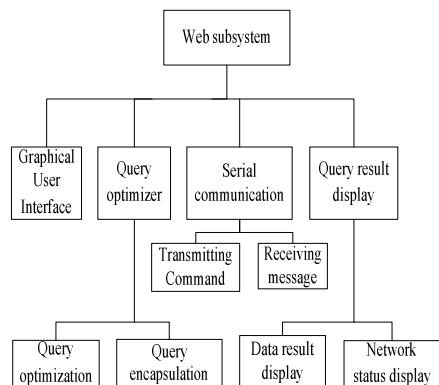


Fig. 2. Module structure of Web subsystem

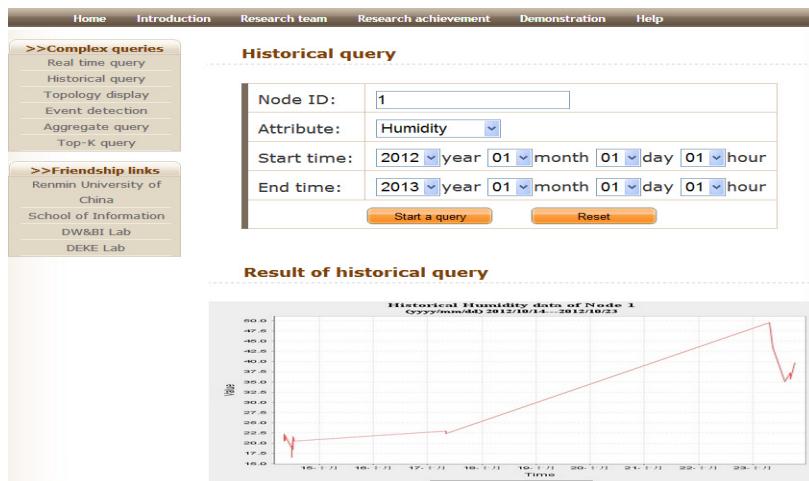


Fig. 3. An example of TinyQP---- Historical query

TinyQP includes two subsystems: the Web subsystem and the query processing subsystem. Next we will describe these subsystems.

2.2 The Web Subsystem

The Web subsystem provides a good user interface for user-customized queries and displays the query result in a graphical way. The Web subsystem uses B/S architecture, and the development language is JAVA. Figure 2 shows the module structure of the Web subsystem. The Web subsystem completes following functions:

(1) Graphical user interface. User can complete the query requirement by browsing the web, clicking the mouse, selecting the drop-down list, etc. The Web subsystem automatically generates SQL-like statement and transmits it to WSNs. The query result will be returned from WSNs and be displayed to the user in the form of charts or graphics. Figure 3 depicts an example of TinyQP---- Historical query.

(2) Query optimizer. The query optimizer analyses the query request, produces the optimized query scheme and forms the query command. The query command includes Command Id, Command Type and Command Content.

(3) Serial communication. The application server sends the query command to the sink node and receives data messages from sink node using the serial communication.

(4) Query result display. The query data will display using dynamic table. The network status, such as network topology, will be shown in a graphical way.

2.3 The Query Processing Subsystem

The sink node accepts command message from the application server via serial communication port, parses command content and executes the corresponding command. Meanwhile, the sink node broadcasts the command message to its neighbors. Neighbors broadcast the command message to their neighbors too, and so on, until all nodes in the network receive the command message.

When ordinary node si receives a command message, si parses command content and performs the query operation based on the Command Type. Assuming the Command Type is Top-k query type, node si extracts the maximum (or minimum) k data from all descendant nodes and itself, and then forwards the Top-k data to the parent node of si , and so on, until reaches the sink node. When the sink node collects the required data, the sink node forwards the data to the application server.

3 Demonstration Settings

TinyQP system deployment environment are jdk1.7/ Tomcat 6.0. Database is MySQL database. The sink node and ordinary nodes are ZigbeX II produced by Hanback Electronics. We demonstrate several complex query algorithms including real time query, historical query, event detection, aggregate query, top-k query, etc. We also demonstrate the network topology in a graphical way.

Acknowledgements. This research was supported by the National Natural Science Foundation of China (61070056, 61033010), the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China (NO.13XNH210).

References

1. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.* 30(1), 122–173 (2005)
2. Yao, Y., Gehrke, J.: Query processing in sensor networks. In: CIDR 2003 (2003)
3. Chow, C., Mokbel, M.F., He, T.: TinyCasper: A Privacy-Preserving Aggregate Location Monitoring System in Wireless Sensor Networks. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1307–1310 (2008)
4. Andreou, P., Zeinalipour-Yazti, D., Vassiliadou, M., Chrysanthis, P.K., Samaras, G.: KSpot: Effectively Monitoring the K Most Important Events in a Wireless Sensor Network. In: 25th IEEE International Conference on Data Engineering, ICDE 2009, pp. 1503–1506 (2009)
5. <http://www.tinyos.net/>

CWePS: Chinese Web People Search

Jianhua Yin and Lili Jiang

Department of Computer Science and Technology,
Tsinghua University, Beijing, China
{jhyin12,juie.jiang}@gmail.com

Abstract. Name ambiguity is a big problem in personal information retrieval, especially given the explosive growth of Web data. In this demonstration, we present a prototype Chinese Web People Search system, called CWePS. Given a personal name as query, CWePS collects the top results from the existing search engines, and groups these returned pages into several clusters. Ideally, the Webpages in the same cluster are related to the same namesake. Specially, we propose a multi-stage strategy to deal with the Chinese personal name disambiguation on the Web and extract some prominent key phrases to describe each namesake.

1 Introduction

Much work has been done on English Web people search [2,4] and this paper presents a Chinese Web people search system. Compared with people search in English, Chinese text processing confronts additional challenges: first, the Chinese text is written in continuous character strings without any word gap in a sentence, which leads more difficulties to the task of tokenization; second, the substrings of a personal name also denote personal names, which is difficult to distinguish; thirdly, there is a higher personal name ambiguity in Chinese than that in other languages. Some related work has been studied, such as the Chinese Personal Name Disambiguation (CPND) campaign[3], which is held to explore personal name disambiguation in Chinese News. However, to the best of our knowledge, few research studies have been done on Chinese Web people search.

To handle these challenges, this paper presents a Chinese Web People Search system called CWePS. Given a personal name as query, we obtain some Web pages from the existing web search engine, and group the Webpages into different clusters using a multi-stage algorithm. On the first stage, we cultivate the context of the namesakes in Knowledge bases (e.g., Baidu Encyclopedia¹), and then combine the Webpages related to the same namesake into new documents after matching the Webpages to these namesakes. On the second stage, we group these new documents and the unmatched Webpages into clusters using the Hierarchical Agglomerative Clustering (HAC) algorithm based on Can't Link (CL) and Must Link (ML) features. An example of the CL features is birthday for which each namesake can only have one instance, and the ML features are unique to each person like personal email address.

¹ <http://baike.baidu.com/>

Our contributions are as follows: (1) we address the Chinese Web people search problem and develop a system, CWePS, to handle its special challenges; (2) we cultivate the Baidu Encyclopedia as well as the entity features to improve the disambiguation performance; (3) CWePS focuses on the Chinese Web People Search problem and gives some intuition for further research.

2 System Architecture

2.1 Preprocessing

Given a personal name, we use a data collector to fetch a list of search results from Google. First, we remove some noise from the results (e.g., the advertisement, HTML and JavaScript codes) using HtmlParser². Second, we employ ICTCLAS³ for the tasks of Chinese word segmentation and part of speech (POS) Tagging. Third, we discard some noisy Webpages, such as the ones without any variants of the queried name and the Webpages in which the queried name is tagged as a non-personal name. In addition, we use Email as ML feature and birthday as CL feature. These features are extracted by regular expression from contexts around the queried name.

We use the vector space model (VSM) to represent the Webpages as $W = \{w_1, w_2, \dots, w_K\}$, in which w_i is the vector of features of the i th Webpage and K is the number of Webpages. We extracted three kinds of features, including nouns, verbs and named entities tagged by ICTCLAS. The weights of the features are measured by Term Frequency (TF) and Inverse Document Frequency (IDF). Compared with a word, the named entity makes more sense for entity (e.g., person) name disambiguation, thus, we assign the higher weights to the named entities.

2.2 Disambiguation

The disambiguation of Chinese personal name is divided into two stages. In the first stage, we extract the descriptions of the namesakes for each queried name from Baidu Encyclopedia as $B = \{b_1, b_2, \dots, b_M\}$, in which M is the number of its namesakes in Baidu Encyclopedia. We calculate the similarity between each Webpage and the description of the namesakes by calculating the cosine similarity [1]. If the similarity is beyond a threshold, we conclude that the Webpage belongs to the corresponding namesake.

In the second stage, we collect the Webpages that are matched into the same namesake in the first stage, then we represent these new documents and the unmatched Webpages as $D = \{d_1, d_2, \dots, d_N\}$ using the vector space model (VSM). We utilize the hierarchical agglomerative clustering (HAC) algorithm [1] based on the CL and ML features to group these new documents into new clusters. Next, we compute the $N \times N$ similarity matrix using Cosine similarity and we further build the CL and ML matrix between these documents. After that, we

² <http://htmlparser.sourceforge.net/>

³ <http://www.ictclas.org/>

execute $N - 1$ steps of merging the currently most similar clusters as long as they don't have different CL features. The algorithm starts with merging the clusters with the same ML features and stops until the similarity between the most similar clusters is below a threshold.

3 System Demonstration

When a user searches “Wei Shen”, the results of CWePS are shown in Figure 1. Each cluster is represented with its most frequent related person, place and organization. When the user clicks each cluster, the Webpages in that cluster will be shown on the right panel. On the left panel of Figure 2, the area of each sector is proportional to the quantity of Webpages of each cluster. When the user clicks any sector, the corresponding Webpages of that cluster will be shown on the right panel, meanwhile the related solid points will be highlighted to show the Webpages' rank distribution in Google search results.

From the results we observe: (1) the combination of the multiple features is useful for personal name disambiguation; (2) the CL and ML features can improve the disambiguation accuracy of personal name disambiguation; (3) some Webpages collected from the social networks (e.g., Weibo.com⁴) have little context but much structured information, which can be used in the future.



Fig. 1. The Result Interface of CWePS

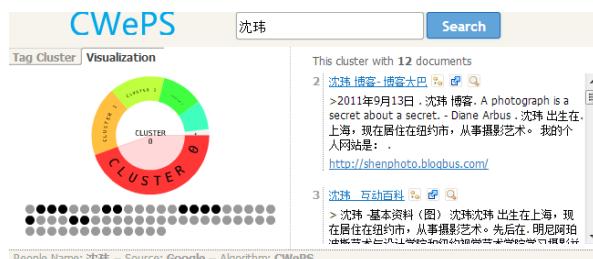


Fig. 2. The Visualization Interface of CWePS

⁴ <http://www.weibo.org/>

Acknowledgement. This work was supported in part by National Natural Science Foundation of China under Grant No. 61272088.

References

1. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press (2008)
2. Jiang, L., Wang, J., An, N.: Grape: A graph-based framework for disambiguating people appearances in web search. In: IEEE International Conference on Data Mining 2009, pp. 199–208 (2009)
3. Ying, C., Jin, P., Li, W.: The Chinese Persons Name Disambiguation Evaluation: Exploration of Personal Name Disambiguation in Chinese News. In: The First Conference on Chinese Language Processing (2010)
4. Javier, A.: Web People Search. PhD Thesis (2009)

CONCERT: A Concept-Centric Web News Recommendation System^{*}

Hongda Ren and Wei Feng

Department of Computer Science and Technology, Tsinghua University
`{rhd10, feng-w10}@mails.tsinghua.edu.cn`

Abstract. A concept is a key phrase which can represent an entity, event or idea that people are interested in. Concept-centric Web news recommendation is a novel content-based recommendation paradigm which can partially alleviate the cold-start problem and provide better recommendation results in terms of diversity than traditional news recommendation systems, as it can capture users' interest in a natural way and can even recommend a new Web news to a user as long as it is conceptually relevant to a main concept of the Web news the user is browsing. This demonstration paper presents a novel CONcept-Centric nEws RecommeDation sysTem called CONCERT. CONCERT consists of two parts: (1) A concept extractor which is based on machine learning algorithms and can extract main concepts from Web news pages, (2) A real-time recommender which recommends conceptually relevant Web news to a user based on the extracted concepts.

1 Introduction

With millions of news generated every day, a high-quality news recommender system is highly desired. However, collaborative filtering technique would fail in this task due to the lack of user feedbacks for real-time news. Although traditional content-based recommender system [1][2][4] can work without feedback, it still has unacceptable drawbacks: (1) A content-based System like Fab [1] represents a Web page as the top 100 most informative words. However, millions of news is published every month, making it both time and space consuming to extract 100 words for every Web page. (2) Other content-based recommender systems such as [2] use TF-IDF [3] as the measurement of “importance” of keywords. These systems often do not consider semantics, thus the recommendation may encounter the topic-drifting problem.

To overcome the above drawbacks, we propose a concept-centric recommender system called CONCERT. A concept here is defined as a key phrase which represents an entity, event, or idea that people are interested in. CONCERT achieves both high diversity and high quality, as its concept-centric methodology can capture users' interest in a natural way. It can also partially alleviate the cold-start problem faced with

* The demonstration system of the CONCERT system is available online at:
<http://dmggroup.cs.tsinghua.edu.cn/CONCERT/main.html>

the traditional recommender systems, since it can even recommend new Web news to a user as long as it is conceptually relevant to the Web news the user is browsing. In [4], Aditya Parameswaran et al. gave a definition of “concept” and developed the so-called K-gram algorithm to extract concepts in English articles, which can be potentially used by CONCERT. However, since the rules used to mine the concepts in K-gram are language-specific, we have to define some rules for each language, which can be very complicated and tedious. As a result, we decide to use a supervised algorithm to extract the main concepts, which is language independent and achieves much better performance.

2 The Framework

The main component of CONCERT is concept extractor and the real-time recommender system.

Concept extractor can extract concepts from the input articles. We build a concept extractor in a supervised way based on the KEA method [5]. It learns five features, that is, TF-IDF, Length, FirstOccur, POST (Part Of Speech Tagging), and FIQL (Frequency In Query Log) from a training set. We use a Naïve Bayesian classifier to train a model which computes the weights of the features. When a new article comes, we can calculate the probability of a phrase being a concept.

After building the concept extractor, we use it to construct the concept database. When users read new Web news, we recommend news to them. Figure 1(a) shows the CONCERT system architecture. Real-time recommender works in 3 steps:

1. Webpage Pre-processing. We extract title and content from our database using boilerpipe¹. After segmentation using Stanford Chinese NLP², we turn urls in our database to news texts expressed as sequences of words.

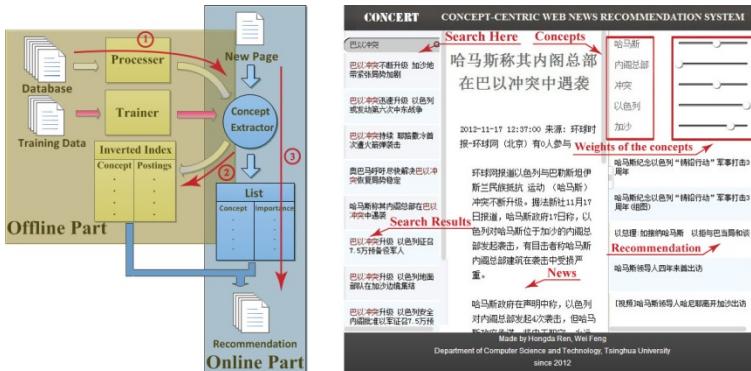


Fig. 1. The Architecture and Interface of CONCERT

¹ <http://code.google.com/p/boilerpipe/>

² <http://nlp.stanford.edu/projects/chinese-nlp.shtml>

2. Concept Indexing. We put all news texts into the concept extractor, and get concepts in every news page in the database. Then we build an inverted index, that for each concept, it contains an inverted list consisting of information about this concept.

3. Real-time Recommender. When a user reads a new Web page, we extract the title and content, segment them, put the news texts into the concept extractor, and then get the main concepts (i.e., the top ranked concepts measured by probability). We then search for each concept in the inverted list and calculate the cosine distance. We put each cosine distance in a minheap of size k to get the top- k results. After calculation we show the main concepts and top- k recommendation results to users.

3 Demonstration Scenario

Our training set contains 13,703 NetEase³ news Web pages used in [6]. Our database contains 3.5 million news Web pages from sohu.com⁴. The interface of the CONCERT system is shown in Figure 1(b).

We plan to demonstrate the CONCERT system with the following scenario:

Step 1. A user searches for news, and the latest news from Internet are displayed.

Step 2. The user clicks a search result and the news is shown in the middle, meanwhile, the concepts and recommendations are shown in the right of the system.

Acknowledgement. This work was supported in part by National Natural Science Foundation of China under Grant No. 61272088.

References

1. Balabanovic, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. Communications of the ACM 40(3), 66–72 (1997)
2. Chen, T., Han, W., Wang, H., Zhou, Y., Xu, B., Zang, B.: Content Recommendation System Based on Private Dynamic User Profile. In: Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, August 19-22 (2007)
3. Salton, G.: Automatic Text Processing. Addison-Wesley (1989)
4. Parameswaran, A., Rajaraman, A., Garcia-Molina, H.: Towards the Web of Concepts: Extracting Concepts from Large Datasets. In: VLDB 2010 (2010)
5. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: Practical Automatic Keyphrase Extraction. In: DL 1999 Proceedings of the Fourth ACM Conference on Digital Libraries, pp. 254–255 (1999)
6. Liu, Z., Chen, X., Zheng, Y., Sun, M.: Automatic Keyphrase Extraction by Bridging Vocabulary Gap. In: Proceedings of the 15th Conference on Computational Natural Language Learning, CoNLL (2011)

³ <http://news.163.com>

⁴ <http://www.sohu.com>

DOI Proxy Framework for Automated Entering and Validation of Scientific Papers

Kun Ma, Bo Yang, and Guangwei Chen

Shandong Provincial Key Laboratory of Network Based Intelligent Computing,

University of Jinan, Jinan, China

{ise_mak,yangbo}@ujn.edu.cn, ujnchengw@gmail.com

Abstract. This paper aims to demonstrate DOI proxy framework for automated entering and validation of scientific papers. This framework, developed in JAVA program, is composed with two components: DOI content service and resolver proxy. After estimating this framework, the metadata of the paper is entered and validated automatically in Web interface. Through a simple presentation, it is partially successful suggesting the framework integrated with third-party systems, such as the management system of the scientific research results and the electronic journal management system.

Keywords: entering, validation, digital object identifier, content negotiation.

1 Introduction

With the increasing of the amount of researchers and its achievements of a scholarly institution, it becomes both more time-consuming for them to enter the scientific paper results. In the meantime, it becomes more egregiously daunting for scientific departments to manually audit the authenticity of each paper. Therefore, a framework of automated entering and validation of the scientific paper would be of great benefit. The challenge is how to enter the bibliography of its scientific paper automatically, and how to ensure the entered bibliography is authentic attached to its owner.

We have classified the available intelligent solutions to this problem, and point out the drawbacks and inefficiencies [1]. The metadata (author, title, source, volume/issue number, starting page number, et al.) extraction of the crawled web page is an approach to paper entering. This approach still needs manual intervention [2]. In non-scanned document extraction approach, the metadata is resolved from the Office Word, L^AT_EX source document and the PDF document conforming to the template. Since the template actually differs from publisher to publisher, this approach needs manual intervention with the high failure rate. Besides the above solutions, there is another approach to look up document details from the document ID system, such as an ISO standard for the document identifier - digital object identifier (DOI) [3]. In 2011, the largest DOI

registry agencies CrossRef and DataCide announced to support unified negotiation called CrossCite content service. However, this approach is not at the more general end of the scale. Some other registration agencies except CrossRef and DataCide do not support content negotiated DOIs. Nowadays, there are still no uniform norms and technical specifications for DOI resolution. Compared with existing approaches, the contribution of our demonstration is that we do some significant development work to propose an integration solution named DOI proxy framework to provide a uniform service.

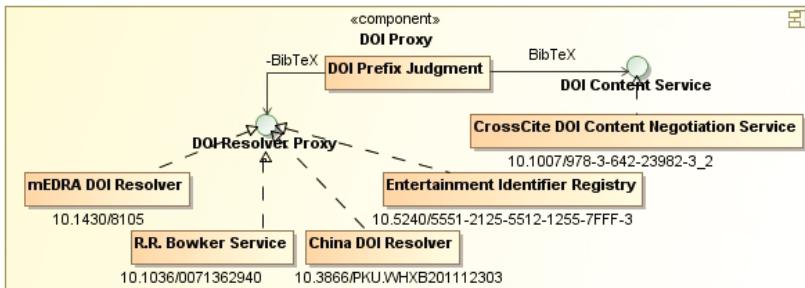


Fig. 1. The architecture of DOI proxy framework

2 DOI Proxy Framework Overview

The architecture of the DOI proxy framework is shown in Figure 1, which is composed of two components: DOI content service and resolver proxy. The request is first redirected to the DOI prefix judgment component to determine the affiliation by the DOI prefix. The contribution of DOI proxy in our framework is that it attempts to solve the heterogeneity problem of the current DOI resolvers, which can also provide a second-development interface for anyone who is interested in it.

DOI Content Service. DOI content service introduces Accept header defined in the HTTP content negotiation to specify certain media types.

DOI Resolver Proxy. DOI resolver proxy provides an interactive and independent interface no matter what the implementation is. For the existing interface of DOI registry agency, we have just integrated it with our framework. For the DOI registry agency without DOI interactive interface, we have implemented DOI resolution first. The specific DOI resolver is just like the flexible plug-in mounting to the proxy.

3 Demonstration Scenarios

We will first use a poster and several slides to introduce the motivation of DOI proxy framework and highlight how it works, which is shown in Figure 2. After



Fig. 2. The screenshot of DOI proxy framework

that, we will show the audience the live demonstration system using HTTP link <http://doi.kunma.net> hosted on our personal website. Finally, we invite them to participate in the interactions to enter the DOI of their paper results by their smart mobile phones or laptops. After estimating this framework, the metadata of the paper is obtained and presented automatically in Web interface. DOI proxy framework compares the result of author name and its affiliation unit with the session context of the integrated system to validate the ownership of the paper.

4 Conclusions

In our demonstration, we will present the DOI proxy framework for automated entering and validation of scientific papers. We will also show the architecture of the core components. As more than 95 percent of DOIs is owned or managed by CrossRef, DataCite, ISTIC and mEDTA DOI registry agencies, this is a common universal approach for the scientific research papers with DOI.

Acknowledgments. This work was supported by the Doctoral Foundation of University of Jinan (XBS1237), and the National Key Technology R&D Program (2012BAF12B07-3).

References

1. Ma, K.: Applying to Literature Metadata Acquisition using DOI and Extraction of Paper Database. *Journal of Modern Information* 32, 44–49 (2012)
2. Wong, T., Lam, W.: Learning to Adapt Web Information Extraction Knowledge and Discovering New Attributes via a Bayesian Approach. *IEEE Transactions on Knowledge and Data Engineering* 22, 523–536 (2010)
3. ISO 26324:2012, *Information and documentation – Digital object identifier system*, http://www.iso.org/iso/catalogue_detail?csnumber=43506

Effectively Return Query Results for Keyword Search on XML Data

Teng He, Guoqing Wu, Qingsong Chen, Junfeng Zhou, and Ziyang Chen

School of Information Science and Engineering, Yanshan University, Qinhuangdao, China

{804743850, 842676860, 1832414806}@qq.com,
{zhoujf, zychen}@ysu.edu.cn

1 Introduction

Recently, XML keyword search has attracted much attention and has become a popular paradigm for information retrieval over XML data. Because of its convenience, users don't need to know a complex query language or the underlying data schema. However, due to the inherent ambiguity, a keyword query usually corresponds to a large number of results that may be classified into different types, or different search intentions, among which only a few meet users' search intentions.

To address this problem, existing methods [1, 3] try to firstly infer users' search intention, i.e., the targeting result type, then return results of that type as query answers. The formula used to compute the score of a result type T , i.e., $C_{T,Q}$, w.r.t. a given keyword query Q is as Formula 1 which is designed based on three guidelines [1], where $A = \log(1 + \prod_{k \in Q} f_k^T)$, k is a keyword of Q , f_k^T is the number of T -typed nodes that contain k in their subtrees; $B = r^{depth(T)}$, r is the reduction factor with range $(0, 1]$, and $depth(T)$ represents the depth of T -typed nodes. Even though [3] has found that this formula suffers from inconsistency and abnormality problems, and made improvement on this formula, in practice, both [1] and [3] still suffer from some of the following problems:

$$C_{T,Q} = A \cdot B \quad (1)$$

1. A node type that is taken as a result type may have many node instances that do not contain all query keywords.

Consider keyword query $Q_1 = \{\text{Tom, XML}\}$ issued on the sample data in Fig. 1. Most likely, it is used to find papers about "XML" written by Tom; hence the result type of Q_1 should be "lab/person/paper". However, [1, 3] take "lab/person" as the result type. Note that node 24 doesn't contain all query keywords.

2. The result type recommended by [1, 3] may contain some query keywords that are descendant of its descendant LCA nodes.

Consider Q_1 again. [1, 3] suggest result type as "lab/person". In fact, node 10 should not be a query result of Q_1 , because after removing the subtree rooted at node 12, the subtree rooted at node 10 does not contain any query keyword of Q_1 , that is, node 10 is not an LCA node.

Besides [1, 3], [2] still cannot work effectively in some cases. Consider query $Q_2 = \{\text{paper, Mike}\}$ issued on the sample data in Fig. 1. Most likely, it is intended to find

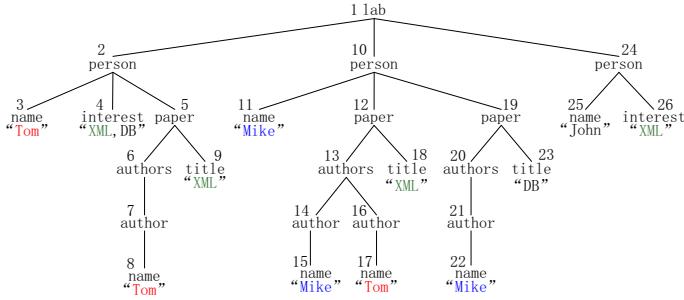


Fig. 1. A sample XML document D

papers written by Mike; hence the result type should be “lab/person/paper”. But [2] still suggest result type as ”lab/person”.

As a comparison, our method avoids all the above problems by using a new ranking function that takes the number of ELCA [4] nodes into account.

2 Overview

2.1 Ranking Function for Result Types

Intuitively, our method takes all ELCA nodes as input to compute the score of each result type, the more the number of ELCA nodes of type T , the more possibility type T be the promising result type of Q . As shown by Formula 2, where S_{ELCA}^T is the set of ELCA nodes of type T .

$$C_{T,Q} = \log |S_{ELCA}^T| \quad (2)$$

2.2 Ranking Function for Results

The main idea of TF*IDF (Term Frequency * Inverse Document Frequency) similarity is: a keyword appearing in many documents should be regarded as being less important than a keyword appearing in a few, while at the same time, a document with more occurrences of a query keyword should be regarded as being more important for that keyword than a document that has less. The formula of TF*IDF similarity is as follows, where Q is the keyword query, d is a document, N is the number of documents, f_k is the number of documents which contain query keyword k , and $f_{d,k}$ is the number of occurrences of keyword k in d .

$$\rho_{Q,d} = \log \left(\sum_{k \in Q} \left(\frac{N}{f_k} \cdot f_{d,k} \right) \right) \quad (3)$$

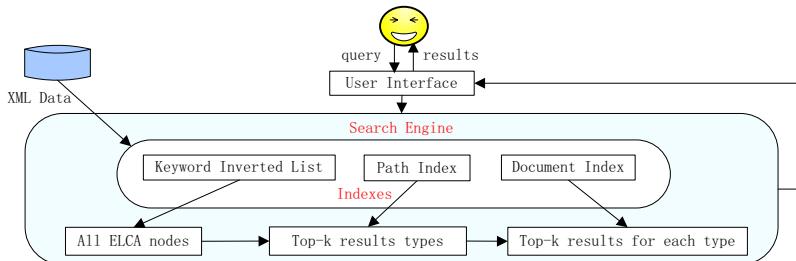


Fig. 2. System architecture

Even though TF*IDF similarity is suitable for flat documents, it only considers the content, regardless of the structure of XML documents. Hence, it cannot be directly applied to XML documents with hierarchical structure. We consider the following rules to compute score of query results.

- Rule 1:** The higher the relevance of a result subtree and a query, the higher the score of the result is.
- Rule 2:** The smaller the distance of query keywords and ELCA node, the higher the score of the result is.
- Rule 3:** The more compact the result subtree, the higher the score of the result is.

The formula of ranking results is shown by Formula 4, where d is a result subtree rooted at an ELCA node, $\rho_{Q,d}$ is the relevance of query Q and d , and $|E_d|$ is the sum of the lengths of the shortest path from each query keyword to the ELCA node. The edges that are repeated on the path are counted only once, which addresses Rule 3.

$$S_{Q,d} = \frac{\rho_{Q,d}}{|E_d| + 1} \quad (4)$$

2.3 Demonstration

Fig. 2 shows the system architecture. The Indexes in Search Engine are created in advance. After user issues a query, we firstly find all ELCA nodes by using Keyword Inverted List, then classify these results into different types according to their node type from Path Index. After that, we compute scores for all these node types by Formula 2 and get top- k node types with the highest score as search intentions to the given keyword query. Finally, for ELCA nodes of each type, we compute scores by Formula 4 and return the first result(the subtree rooted at ELCA node) of each result type to users. The Document Index records the informations contained in each node.

Acknowledgment. This research was partially supported by the National Natural Science Foundation of China (No. 61073060, 61272124), the Research Funds from Education Department of Hebei Province (No. Y2012014), and the Science and Technology research and development program of Hebei Province (No. 11213578).

References

1. Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective XML keyword search with relevance oriented ranking. In: ICDE, pp. 517–528 (2009)
2. Li, J., Liu, C., Zhou, R., Wang, W.: Suggestion of promising result types for XML keyword search. In: EDBT, pp. 561–572 (2010)
3. Li, J., Wang, J.: Effectively inferring the search-for node type in XML keyword search. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010, Part I. LNCS, vol. 5981, pp. 110–124. Springer, Heidelberg (2010)
4. Xu, Y., Papakonstantinou, Y.: Efficient LCA based keyword search in XML data. In: EDBT, pp. 535–546 (2008)

BrackitMR: Flexible XQuery Processing in MapReduce

Caetano Sauer, Sebastian Bächle, and Theo Härdter

University of Kaiserslautern
P.O. Box 3049, 67653 Kaiserslautern, Germany
`{csauer,baechle,haerder}@cs.uni-kl.de`

Abstract. We present BrackitMR, a framework that executes XQuery programs over distributed data using MapReduce. The main goal is to provide flexible MapReduce-based data processing with minimal performance penalties. Based on the Brackit query engine, a generic query compilation and optimization infrastructure, our system allows for a transparent integration of multiple data sources, such as XML, JSON, and CSV files, as well as relational databases, NoSQL stores, and lower-level record APIs such as BerkeleyDB.

1 Introduction

The success of query interfaces for MapReduce (MR), such as Hive [4] and Pig [1], indicates the importance of having simple distributed data-processing middlewares, like MR, combined with the expressive power and ease of use of query-processing languages. A breakthrough of such systems is the flexibility provided in comparison with traditional data warehouses and parallel databases. The need for flexibility is particularly strong in the *Big Data* domain, where data is frequently processed in an ad-hoc manner and usually does not follow a strictly relational, previously known schema.

Our approach, based on XQuery, takes the flexibility of MR-based query processing one step further. The starting point is the Brackit query engine, which compiles and optimizes XQuery programs in a storage-independent manner. Its data model extends the original XQuery data model, allowing the system to natively process not only XML structures, but also JSON and relational data. This allows us to plug-in various data sources that reuse the generic compilation and optimization steps performed by the engine. Thanks to a well-defined interface between query processing and storage layers, storage-related optimizations such as predicate and projection push-down are also exploited, thereby minimizing the performance penalty that usually comes as a trade-off for flexibility.

BrackitMR is a component which extends the Brackit query engine to produce MR jobs, transparently benefiting from the optimization techniques and the versatile storage integration. As we demonstrate, such capabilities go beyond the simple raw-file approach of related systems like Pig and Hive, providing wider applicability and better integration with existing infrastructures like DBMSs and data-warehouse systems.

2 System Overview

Our system is composed of three main components: (i) the Brackit engine, which not only performs query compilation and optimization, but is also responsible for the execution of physical query plans within a single node; (ii) the BrackitMR component, which translates query plans into equivalent MR job specifications; and (iii) the Collection component, which manages multiple data sources that implement a unified interface for retrieving data, mapping into our extended data model, and performing push-down of operations like filters and projections.

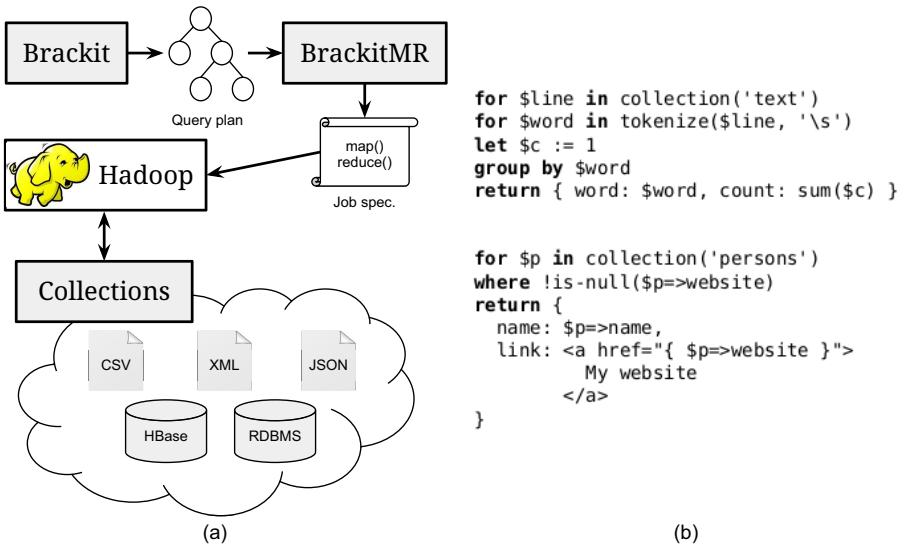


Fig. 1. (a) BrackitMR architecture and (b) a sample query accessing multiple stores

Figure 1a illustrates the three components of our system and how they interact with each other and with the Hadoop system. A query is first compiled and optimized by the Brackit engine, generating a logical query plan which is processed by the BrackitMR component. It splits the query plan into blocks of non-blocking operators which are assigned to Map and Reduce functions, as described in [2]. The technique applied is the same in systems like Hive [4] and Pig [1]. The generated job specifications are then submitted to the Hadoop cluster. When the Map and Reduce functions are executed, the parts of the logical plan contained within them are finally translated into physical operators by a local instance of the Brackit engine. When executed, these operators fetch data from the Collections component, which transparently handles a variety of data sources, producing either XML, JSON, or relational data. Figure 1b shows two sample queries. The top one implements the classical *word count* problem, while

the second one reads data from a JSON data source and produces records containing embedded XML, which is made possible by our extended data model. Both queries produce JSON records as output, which can be either printed on the screen or written to a data format specified by the user in the configuration. Note the use of the `=>` operator, which extracts a field from a JSON record.

3 Demo Description

The system we demonstrate consists of a Web interface which submits queries to a BrackitMR server. The attendees will be able to select from a predefined set of queries as well as to write their own queries. The interface will display the query plans and the steps of transforming it into a MR job. Execution will be monitored and the user will have the option to activate tracing capabilities and debug flags to investigate how data is transformed and shipped across nodes in the Hadoop cluster. The data sources provided will include plain text files, CSV, JSON, relational databases, native XML databases, BerkeleyDB container files, HBase tables, MongoDB collections, and so on. The unified interface of the Collection component allows such sources to be transparently processed within the same query, allowing data from heterogeneous sources to be processed and combined into a single dataset.

Another feature of our interface is benchmarking, which gives the possibility to compare query execution times with optimization features turned on and off. Furthermore, it is possible to compare the performance of the same query running on data stored in different formats, as well as compare the performance of BrackitMR with that of Pig and Hive.

Besides the ability to process data from various sources, BrackitMR is based on the XQuery language, which has a higher expressive power than PigLatin and HiveQL. As discussed in [3], XQuery also fulfills a wider set of flexibility requirements—particularly in the Big Data scenario—than SQL and the mentioned related approaches. Our extended data model, which adds support for JSON, also makes BrackitMR a perfect fit for scenarios in which XML is too cumbersome, but a flexible nested data model is still desired.

References

1. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A Not-So-Foreign Language for Data Processing. In: SIGMOD Conference, pp. 1099–1110 (2008)
2. Sauer, C.: XQuery Processing in the MapReduce Framework. Master’s thesis, University of Kaiserslautern, Germany (2012)
3. Sauer, C., Härdter, T.: Compilation of Query Languages into MapReduce. *Datenbank-Spektrum* 13(1), 5–15 (2013)
4. Thusoo, A., Sen Sarma, J., Jain, N., Shao, Z., Chakka, P., Zhang, N., Anthony, S., Liu, H., Murthy, R.: Hive – A Petabyte Scale Data Warehouse using Hadoop. In: ICDE Conference, pp. 996–1005 (2010)

Author Index

- Amagasa, Toshiyuki 111
An, Ning 58
- Bächle, Sebastian 806
Bai, Xue 37
Berti-Equille, Laure 64
- Cameron, Juan J. 51
Chen, Changbing 118
Chen, Guangwei 799
Chen, Hong 282, 788
Chen, Hongmei 326
Chen, Huahui 741
Chen, Jianwen 216
Chen, Jimeng 666
Chen, Lu 429
Chen, Mingming 717
Chen, Ming-Syan 533
Chen, Qingsong 802
Chen, Sixi 58
Chen, Wei 289
Chen, Zhong 564, 729
Chen, Ziyang 472, 802
Cheng, Chao 326
Cheng, Hong 1
Cheng, Pang 423
Ci, Xiang 142
Cui, Bin 769
Cuzzocrea, Alfredo 51
- Dai, Ke 173
Dapeng, Lang 757
Dong, Qixiang 729
Dong, Xin Luna 64
Du, Liang 241, 253
- Fan, Yulei 382, 393
Fang, Ying 583
Fang, Zhiwen 717
Feng, Chong 583
Feng, Huabin 77
Feng, Jianhua 203
Feng, Ling 216
Feng, Wei 796
Feng, Xing 190
- Feng, Yansong 781
Feng, Zhiyong 570
- Gao, Hong 77, 87, 351, 514
Gao, Kening 521
Gao, Yang 99
Gao, Yunjun 190, 429
Gong, Weihua 417
Gong, Xi 643
Guan, Jihong 465
Guan, Zhi 729
Guo, Longjiang 453
Guo, Rui 351
- Härder, Theo 806
Han, Qilong 595
Hassanlou, Nasrin 545
He, Bingsheng 118
He, Jia 631
He, Teng 802
He, Xianmang 741
Hlaing, Aye Thida 229
Htoo, Htoo 229
Hu, Jianbin 564
Hu, Qingcheng 99
Hu, Xiangyu 607
Huang, Heyan 583
Huang, Lian'en 338, 363
Huang, Lifu 363
Huang, Yalou 666
- Inoue, Hiroyuki 111
- Jain, Rohit 440
Jia, Aixia 781
Jiang, Dongming 649
Jiang, Fan 51
Jiang, Fei 13
Jiang, Jing 502
Jiang, Lili 792
Jiang, Longxiang 570
Jiang, Tao 429
Jiang, Yuan 649
- Jin, Lian 87
Jin, Peiquan 613
Jin, Ruoming 1

- Kitagawa, Hiroyuki 111, 301
 Lai, Wenyu 382
 Lanju, Kong 423
 Lee, Adam J. 643
 Lei, Kai 289
 Leung, Carson K. 51
 Li, Cuiping 282, 788
 Li, Dong 741
 Li, Fan 631
 Li, Guoliang 179
 Li, Hongjie 338, 363
 Li, Jianzhong 77, 351, 514, 595
 Li, Jinbao 453
 Li, Kaiyu 351
 Li, Lian 58
 Li, Pengfei 25
 Li, Pengyuan 595
 Li, Qingzhong 13
 Li, QiuHong 173
 Li, Rui 130
 Li, Sujian 502
 Li, Tianrui 631
 Li, Xiaoming 314
 Li, Xuhui 105
 Li, Yujia 741
 Li, Zhoujun 655, 691
 Lian, Xin 607
 Liao, Lejian 698
 Lin, Huaizhong 25
 Lin, Sheng 613
 Lin, Yu-Chieh 533
 Liu, An 631
 Liu, Chunyang 655, 691
 Liu, Jie 666
 Liu, Mengchi 105
 Liu, Mingchao 417
 Liu, Peng 289
 Liu, Qing 429
 Liu, Qizhi 167
 Liu, Yueqin 289
 Liu, Zheng 1
 Liu, Zhiyuan 478
 Lu, Dongming 25
 Lü, Kevin 326
 Lu, Yang 314
 Lv, Yanfei 769
 Ma, Baojun 332
 Ma, Di 698
 Ma, Kun 799
 Ma, Pengfei 99
 Ma, Youzhong 155
 Meng, Xiaofeng 142, 155, 382, 393, 557, 705
 Mo, Shangfeng 788
 Mohania, Mukesh K. 440
 Ohsawa, Yutaka 229
 Ou, Gaoyan 289
 Pan, Haiwei 595
 Pan, Yifan 417
 Pei, Bin 282
 Peng, Shanglian 631
 Peng, Wu 757
 Peng, Zhaohui 13
 Prabhakar, Sunil 440
 Qingzhong, Li 423
 Rao, Junyang 781
 Ren, Hongda 796
 Ren, Jiangtao 265
 Ren, Meirui 453
 Sakauchi, Masao 229
 Sauer, Caetano 806
 Shaikh, Salman Ahmed 301
 Shao, Yingxia 769
 Shaobin, Huang 757
 Shen, Xukun 375, 747
 Shen, Yi 679
 Shen, Yi-Dong 241, 253
 Shen, Zhiyong 253
 Shi, Xiaogang 769
 Shoaran, Maryam 545
 Sonehara, Noboru 229
 Song, Dandan 698
 Song, Wei 619
 Srivastava, Divesh 64
 Stannus, Simon 417
 Su, Mingchuan 405
 Sun, Lin 58
 Sun, Maosong 478
 Sun, Weiwei 173
 Tang, Jiayu 478
 Tang, Xiaolong 570
 Tao, Chengkai 167

- Thomo, Alex 545
Tian, Feng 375, 747
Tianyang, Lv 757

Wang, Bo 472
Wang, Guoren 521
Wang, Haiping 142
Wang, Hongda 619
Wang, Hongjun 631
Wang, Hongzhi 77, 87, 351, 514
Wang, Huazhong 25
Wang, Jianying 253
Wang, Miao 705
Wang, Peng 173
Wang, Qian 13
Wang, Shan 130, 235, 405
Wang, Tengjiao 289
Wang, Tiejun 631
Wang, Wei 173, 741
Wang, Xin 570
Wang, Xinjing 453
Wang, Xue 405
Wang, Xun 502
Wang, Yang 173, 203
Wang, Yanhao 130
Wang, Yuan 666
Wang, Zhihui 173
Wei, Qiang 332
Wei, Qingting 465
Wei, Xiaochi 583
Wu, Guoqing 802
Wu, Junjie 717
Wu, Liang 729
Wu, Xiaohong 277
Wu, Yongwei 58
Wu, Zhengang 564

Xie, Hui 514
Xie, Jianjun 679
Xie, Wuping 649
Xie, Xiaoqin 595
Xin, Xin 583
Xing, Chunxiao 99
Xing, Jianchun 619
Xiong, Yun 37
Xu, Ke 717
Xu, Zhiwu 253

Yan, Hongfei 314
Yang,Bo 799

Yang, Dongqing 289
Yang, Liang Huai 417
Yang, Qiliang 619
Yang, Tao 564
Yang, Wenzhong 277
Yang, Xianxiang 583
Yin, Jianhua 792
Yin, Yanshen 99
You, Zhen 649
Yu, Ge 521
Yu, Jeffrey Xu 1
Yu, Jianjun 679
Yu, Liangwen 564
Yu, Philip S. 533
Yu, Ting 643
Yuan, Hua 332
Yuan, Xiaojie 607
Yue, Lihua 613

Zhang, Bin 429
Zhang, Chaogui 265
Zhang, Ende 521
Zhang, Haijun 655, 691
Zhang, Haiwei 607
Zhang, Hao 490
Zhang, Jinzeng 557
Zhang, Jun 216
Zhang, Nan 203
Zhang, Qingqing 613
Zhang, Wenjia 619
Zhang, Wenjie 190
Zhang, Xiao 130
Zhang, Xiaojian 705
Zhang, Xiaoming 655, 691
Zhang, Xiaoxiang 613
Zhang, Xiaoying 788
Zhang, Xuewei 619
Zhang, Yan 167
Zhang, Yansong 235, 405
Zhang, Ying 190
Zhang, Yong 99
Zhang, Yu 155
Zhang, Zhen 277
Zhang, Zhenyu 277
Zhang, Zhilin 25
Zhang, Zhiming 338
Zhang, Zhiqiang 595
Zhao, Dongyan 781
Zhao, Jichang 717
Zhao, Jinjia 472

- Zhao, Suyun 282
Zhao, Tingting 282
Zhao, Wayne Xin 314
Zhao, Xiang 190
Zhao, Xingmin 472
Zheng, Weimin 58
Zhong, Ming 105
Zhou, Junfeng 472, 802
Zhou, Lihua 326
Zhou, Wenjun 490
- Zhou, Xin 130
Zhou, Xuan 235, 405
Zhu, Feida 502
Zhu, Hengshu 37
Zhu, Jiawei 564
Zhu, Shanfeng 105
Zhu, Yangyong 37
Zhu, Yuean 235
Zhuang, Kun 583