

LNCS 10708

Riccardo Guidotti · Anna Monreale
Dino Pedreschi · Serge Abiteboul (Eds.)

Personal Analytics and Privacy

An Individual and Collective Perspective

First International Workshop, PAP 2017
Held in Conjunction with ECML PKDD 2017
Skopje, Macedonia, September 18, 2017
Revised Selected Papers



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7409>

Riccardo Guidotti · Anna Monreale
Dino Pedreschi · Serge Abiteboul (Eds.)

Personal Analytics and Privacy

An Individual and Collective Perspective

First International Workshop, PAP 2017
Held in Conjunction with ECML PKDD 2017
Skopje, Macedonia, September 18, 2017
Revised Selected Papers

Editors

Riccardo Guidotti 
KDDLab, ISTI-CNR
Pisa
Italy

Anna Monreale 
KDDLab, University of Pisa
Pisa
Italy

Dino Pedreschi
KDDLab, University of Pisa
Pisa
Italy

Serge Abiteboul
Inria, École Normale Supérieure
Paris
France

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-3-319-71969-6

ISBN 978-3-319-71970-2 (eBook)

<https://doi.org/10.1007/978-3-319-71970-2>

Library of Congress Control Number: 2017960859

LNCS Sublibrary: SL3 – Information Systems and Applications, incl. Internet/Web, and HCI

© Springer International Publishing AG 2017

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.

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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature

The registered company is Springer International Publishing AG

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The First International Workshop on Personal Analytics and Privacy (PAP) was held in Skopje, Macedonia, on September 18, 2017. The purpose of the workshop is to encourage principled research that will lead to the advancement of personal data analytics, personal services development, privacy, data protection, and privacy risk assessment with the intent of bringing together researchers and practitioners interested in personal analytics and privacy. The workshop, collocated with the conference ECML/PKDD 2017, sought top-quality submissions addressing important issues related to personal analytics, personal data mining, and privacy in the context where real individual data (spatio temporal data, call details records, tweets, mobility data, transactional data, social networking data, etc.) are used for developing data-driven services, for realizing social studies aimed at understanding nowadays society, and for publication purposes.

The authors were invited to submit original research or position papers proposing novel methods or analyzing existing techniques on novel datasets on any relevant topic including, but are not limited to, the following:

- Personal model summarizing the user’s behaviors
- Personal data and knowledge management (databases, software, formats)
- Personal data collection (crawling, storage, compression)
- Personal data integration
- Personal data store and personal information management systems models
- Parameter-free and auto-adaptive methodologies for personal analytics
- Novel indicators measuring personal behavior
- Individual vs. collective models
- Privacy-preserving mining algorithm
- Privacy-preserving individual data sharing
- Privacy risk assessment
- Privacy and anonymity in collective services
- Information (data/patterns) hiding
- Privacy in pervasive/ubiquitous systems
- Security and privacy metrics
- Personal data protection and law enforcement
- Balancing privacy and quality of the service/analysis
- Case study analysis and experiments on real individual data

All submitted papers were reviewed on the basis of technical quality, relevance, significance, and clarity by at least three referees. The Program Committee received 19 submissions and accepted 13 papers. The program of PAP was enriched by the keynote speeches by Bruno Lepri entitled “The Rise of Decentralized Personal Data Markets” and by Serge Abiteboul entitled “Personal Knowledge Management Systems.”

We would also like to thank the Program Committee for their work on reviewing the papers. The process of reviewing and selecting papers was significantly simplified through using EasyChair. We thank all attendees to the workshop and hope that this event will enable a good exchange of ideas and generate new collaborations among attendees. The organization of PAP 2017 was supported by the European Community’s H2020 Program under the funding scheme “INFRAIA- 1-2014–2015: Research Infrastructures” grant agreement 654024, <http://www.sobigdata.eu>, “SoBigData: Social Mining & Big Data Ecosystem.”

October 2017

Riccardo Guidotti
Anna Monreale
Dino Pedreschi
Serge Abiteboul

Organization

Program Committee

Nicolas Anciaux	Inria, France
Bettina Berendt	KU Leuven, Belgium
Elisa Bertino	Purdue University, USA
Tobias Blanke	King's College London, UK
Francesco Bonchi	ISI Foundation, Italy
Paolo Cintia	ISTI-CNR, Italy
Michele Coscia	Harvard University, USA
Mathieu Cunche	INSA-Lyon/Inria, France
Jon Crowcroft	University of Cambridge, UK
Boxiang Dong	Montclair State University, USA
Wendy Hui Wang	Stevens Institute, USA
Bruno Lepri	MobS Lab at Fondazione Bruno Kessler, Italy
Mirco Musolesi	University College London, UK
Francesca Pratesi	University of Pisa, Italy
Vincenc Torra	IIIA-CSIC, Spain
Jeroen Van Der Hoven	Delft University, The Netherlands
Michele Vescovi	Telecom Italia, Italy

Introduction of the Editors

Personal Analytics and Privacy

An Individual and Collective Perspective

Riccardo Guidotti¹ and Anna Monreale²

¹ ISTI-CNR, Via G. Moruzzi, 1, Pisa

Riccardo.Guidotti@isti.cnr.it

² University of Pisa, Largo B. Pontecorvo, 3, Pisa

Anna.Monreale@di.unipi.it

1 We All Need to Own and Use Our Own Data

Every year, each person leaves behind her more than 5 GB of *digital breadcrumbs*, disseminated by disparate systems that we use for our daily activities: traveling, communicating, paying for goods, banking, searching the web, listening music, reading, playing, posting or tweeting, screening our health. Five gigabytes, without taking into account photos and videos, otherwise numbers would grow considerably. An avalanche of personal information that, in most cases, gets lost. Only each single individual could connect all this personal information into some personal data repository. No Google or Facebook has a similar power today, and we should very carefully avoid this possibility in the future. The fact that in the contemporary initial phase of a measurable society there are few large harvester, or “latifundists”, who store data on masses of people in large inaccessible repositories in an *organization-centric* model, does not mean that *centralization* is the only possible model, nor the most efficient and sustainable.

Nowadays, data and information belong to big organizations (Amazon, Google, Facebook, etc.) which employ *top-down* control over these data. They can create a mosaic of human behaviors used to extract valuable knowledge for marketing purposes: *our personal data is the new gold*. For example, users produce personal data like Facebook posts, or GPS movements using Google Maps, or online shopping through Amazon, and these data are collected and obscurely employed by these companies for marketing or to produce services. On the other hand, individuals do not have the tools and capabilities to extract useful knowledge from their personal data. This is a *Legrand Star* model [11], i.e., a centralized network model, where users can not directly control and exploit their own personal data. Data owning and usage would require not a *bottom-up* system, but a *Baran Web* model, i.e., a *peer distributed approach*, a network of peers, both individual and companies, in which no single node has absolute control of everything but everyone controls thyself, and has only a partial vision of the surrounding peers. The first brick that must be placed to build this *Web* and to start a change of perspective, is the development of *Personal Data Models*, which are sewn on each individual to fit their subjective behaviors.

Data Mining applied to individual data creates an invaluable opportunity for individuals to improve their *self-awareness*, and to enable *personalized services*. However, nowadays users have a limited capability to exploit their personal data, thus we need a change of perspective towards a *user-centric* model for personal data management: a vision compatible with the data protection reform of EU, and promoted by the World Economic Forum [12, 14, 15].

2 Making Sense of Own Personal Big Data

Although some user-centric models like the *Personal Intelligent Management Systems (PIMS)* and the *Personal Data Store (PDS)* are emerging [1, 5], currently there is still a significant lack in terms of algorithms and models specifically designed to capture the knowledge from individual data and to ensure privacy protection in a user-centric scenario.

Personal data analytics and individual privacy protection are the key elements to leverage nowadays services to a new type of systems. The availability of personal analytics tools able to extract hidden knowledge from individual data while protecting the privacy right can help the society to move from organization-centric systems to user-centric systems, where the user is the owner of her personal data and is able to manage, understand, exploit, control and share her own data and the knowledge deliverable from them in a completely safe way.

Recent works are trying to extend the user-centric models for data management with tools for *Personal Data Analytics* [8]. With *Personal Data Analytics* are indicated the personal data mining processes extracting the user models, and providing self-awareness and personalized services. Personal Data Analytics can be exploited (*i*) to improve the user *self-awareness* thanks to the personal patterns they unveil, and (*ii*) to empower *personalized services* by providing proactive predictions and suggestions on the basis of the user's profile.

In practice, Personal Data Analytics allows a user to make sense of her personal data and to exploit it [9]. Figure 1(a) shows the overall Personal Data Analytics approach. The individual data flow into the PDS and are stored according to one of the possible technique described in the PDS literature [1, 4, 5]. Along the analysis of the continuous digital breadcrumbs, the PDS must consider that it does not exist a unique and constant model describing human behaviors. Indeed, our behaviors will be never “in equilibrium” because we constantly move, we buy new things, we interact with our friends, we listen to music, etc., generating in this way a non-interruptible flow of personal data [2]. Therefore, Personal Data Analytics must be *dynamic* and *adaptable* to continuous changes. The user profile described by Personal Data Analytics can be used to improve the user *self-awareness* and for *personalized services* yet adopting Personal Data Analytics. Self-awareness can be provided for example through a dashboard where the user can navigate and understand her models and patterns. Examples of personalized services can be *recommendation systems* or *predictors* of future actions.

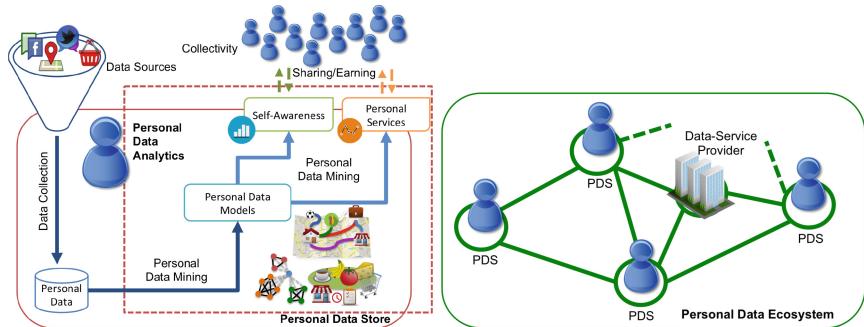


Fig. 1. (a) Personal Data Analytics (left); (b) Personal Data Ecosystem (right).

3 The Personal Data Ecosystem and the Privacy Issues

The PDS extended with PDM offers an augmented image of ourselves, our image reflected in a *digital mirror*. However, passive personal data collection and knowledge mining need to be balanced with *participation*, based on a much greater awareness of the value of own personal data for each one of us and the communities that we inhabit, at all scales.

Personal Data Analytics enables the comparison of our individual patterns with the collective ones of the communities we belong to, provided we have a way to interact and collaborate with other peers, individuals and institutions that are, in turn, equipped with their PDS's and connected to each other in a network. Figure 1(a) (top right) shows how, to provide and obtain improved *self-awareness* and *personalized services*, a user can share information and, at the same time, earn knowledge, by communicating with the *collectivity*.

This enables a *Personal Data Ecosystem (PDE)* (Fig. 1(b)) that can be modeled as distributed network of peers, both individual users and public or private institutions and companies, each one with their own PDS and PDM [8]. The PDE can generate an innovative form of *collective awareness*, characterized by a self-reinforcing loop where (i) superior individual knowledge is created by comparison with collective knowledge, enhancing individual ability to better align own goals with common interest, and (ii) superior collective knowledge is created through the active participation of individuals in a decentralized system, i.e., without centralization of unnecessarily large amounts of information. The PDE is in line with the *peer progressive* idea of a decentralized network where news, money, and knowledge come from the periphery instead of from the center [11]. In [10] it is compared the concept of “wise king” (centralized system) against the Adam Smith’s “invisible hand” regulating a decentralized self-organizing system. Furthermore, the PDE idea outlines the *Nervousnet* project: a globally distributed, self-organizing, techno-social system for answering analytical questions about the status of world-wide society, based on social sensing, mining and the idea of trust networks and privacy-aware social mining [7].

Although the PDE setting enables new opportunities for individuals who may exploit their own data to improve the daily life with more and more self-awareness, it is

impossible to ignore the possible privacy risks that may derive from the sharing of personal data and the knowledge extractable from them. Indeed, the worrying aspect of this story is that often, individual data provide a very fine detail of the individual activities and thus, in case of *sensitive* activities the opportunities of discovering knowledge increase with the risks of *privacy violation*. The threat goes as far to recognize personal or even sensitive aspects of their lives, such as home location, habits and religious or political convictions. Managing this kind of data is a very complex task, and often we cannot solely rely on de-identification (i.e., removing the direct identifiers contained in the data) to preserve the privacy of the people involved. In fact, many examples of re-identification from supposedly anonymous data have been reported in the scientific literature and in the media, from health records to GPS trajectories and, even, from movie ratings of on-demands services. Several techniques have been proposed to develop technological frameworks for countering privacy violations, without losing the benefits of big data analytics technology [6]. Unfortunately, most of the research work done in the context of privacy-preserving data mining and data analytics focuses on an organization-centric model for the personal data management, where the individuals have a very limited possibility to control their own data and to take advantage of them according to their needs and wills. PDE instead provides to individuals the chance to have a central and active role in the control of the lifecycle of her own personal data introducing also a layer of transparency. In particular, it enables individuals to control a copy of their data and/or the knowledge extracted from them. In practices, the individuals acquire the right to dispose or distribute their own individual information with the desired privacy level in order to receive services or other benefits or in order to increase their knowledge about themselves or about the society they live in. In this setting to guarantee the information sharing with the level of desired privacy level, *Privacy-by-Design* data transformations [3, 13] must be applied before data leave the user. This guarantees the prevention of privacy attacks to the PDE addressing possible privacy issues with a pro-active approach. This encourages the voluntary participation of users, limiting the fear and skepticism that often leads people to not access the benefits of extracting knowledge from their own data, both at personal and collective level.

Acknowledgments. This work and workshop has been partially supported by the European Community’s H2020 Program under the funding scheme ‘INFRAIA-1-2014–2015: Research Infrastructures’ grant agreement 654024, ‘SoBigData: Social Mining & Big Data Ecosystem’ (<http://www.sobigdata.eu>).

References

1. Abiteboul, S., André, B., Kaplan, D.: Managing your digital life. *Commun. ACM*, **58**(5), 32–35 (2015)
2. Ball: Why Society is a Complex Matter. Springer (2012)
3. Cavoukian: Privacy design principles for an integrated justice system. TR (2000)

4. Vescovi, M., et al.: My data store: toward user awareness and control on personal data. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, pp. 179–182 (2014)
5. de Montjoye, Y.-A., Shmueli, E., Wang, S.S., Pentland, A.S.: openpds: protecting the privacy of metadata through safeanswers. *PloS one* **9**(7), e98790 (2014)
6. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: a survey of recent developments. *ACM Comput. Surv.* **42**(4), 14:1–14:53 (2010)
7. Giannotti, et al.: A planetary nervous system for social mining and collective awareness. *Eur. Phys. J. Spec. Top.* **214**(1), 49–75 (2012)
8. Guidotti, R.: Personal data analytics: capturing human behavior to improve self-awareness and personal services through individual and collective knowledge (2017)
9. Guidotti, R., et al.: Towards user-centric data management: individual mobility analytics for collective services. In: *ACM SIGSPATIAL*, pp. 80–83 (2015)
10. Helbing: The automation of society is next: how to survive the digital revolution. Available at SSRN 2694312 (2015)
11. Johnson: Future Perfect: The Case for Progress in a Networked Age. Penguin (2012)
12. Kalapesi: Unlocking the value of personal data: from collection to usage. In: *WEF* (2013)
13. Monreale, A., Rinzivillo, S., Pratesi, F., Giannotti, F., Pedreschi, D.: Privacy-by-design in big data analytics and social mining. *EPJ Data Sci.* **3**(1), 10 (2014)
14. Pentland, A., et al.: Personal data: the emergence of a new asset class. In: *WEF* (2011)
15. Rose, et al.: Rethinking personal data: strengthening trust. In: *WEF* (2012)

Abstracts of Invited Talks

The Emergence of Personal Data Markets

Bruno Lepri

Fondazione Bruno Kessler, 38123 Trento, Italy

Abstract. The almost universal adoption of mobile phones, the exponential increase in the use of Internet services and social media platforms, and the proliferation of wearable devices and connected objects have resulted in a massive collection of personal data that represents an invaluable resource for designing and building systems able to understand the needs and activities of both individuals and communities - so as to provide personalized, context-aware services. Hence, many public services of societal interest (e.g. emergency response, city planning, etc.) are dependent on this data. At the same time, many commercial services require sensitive data such as location, purchase, or browsing history. However, this scenario raises unprecedented privacy challenges derived from the collection, storage and usage of vast amounts of personal data. The core problem is how can an individual share sensitive data for a public service or a desired commercial service and be sure that the data will only be used for the intended purpose? This question implicitly recognizes the risks in terms not only of possible abuses in the usage of the personal data but also of the “missed chance for innovation” that is inherent to the current dominant paradigm of siloed data collection, management, and exploitation, which precludes participation to a wide range of actors, most notably the very producers of personal data (i.e. the users). Recently, new user-centric models for personal data management have been proposed, in order to provide individuals with more control of their own data’s life-cycle. To this end, researchers and companies are developing repositories which implement medium-grained access control to different kinds of personally identifiable information (PII). Previous work has also introduced the concept of personal data markets in which individuals sell their own personal data to entities interested in buying it. While personal data markets might be the next natural step in designing technology to support a transparent and privacy-preserving use of personal data, they are still at a research stage due to a number of technological and regulatory challenges. In order to implement a market, it is necessary to connect potential buyers (demand) with sellers (offer) and provide a trustworthy mechanism for the exchange of goods or value between buyers and sellers. In a personal data market, the sellers are individuals (who own their personal data), the buyers are corporations, researchers, governments, etc., and the mechanism for the exchange of “goods” is still to be defined. In my talk, I provide a possible answer to such issue describing recent efforts to develop a user-centric Personal Data Market approach, coupled with cryptographic guarantees on data access and usage.

Personal Knowledge Base Systems

Serge Abiteboul and David Montoya

Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL
Research University, 75005 Paris, France

The typical Internet user has personal data spread over several devices and across several online systems. Geeks already know how to control their personal data. It is now (or soon will be) possible for everyone to do the same, and there are many advantages to doing so. Everyone should now manage his/her personal info management system. We explain the main features of *personal information management systems* (sometimes called, self-data management systems). We consider advantages they provide from societal viewpoints. We argue that they are coming because of a number of reasons, political, economical, technical, and societal.

We believe that a main argument for their future success is that they enable new functionalities based on knowledge integration, and thus that the future systems are personal *knowledge* management systems. Such a system should integrate into a coherent whole data of very different nature, in particular, emails and other messages, calendars, contacts, GPS locations, web searches, bills, bank and medical records, etc. The integration of data that have typically to be exported from various systems should be performed by transforming all the information into a single knowledge base on a machine the user controls. The resulting system thus acts as a digital home for all the digital knowledge of the person.

This integration allows the user to query her personal information within and across different dimensions. It also enables performing analytics to learn from all this information. We believe that many users would be reluctant to give access to all their information to a single company, or if they do that, they will require strong guarantee of privacy. The fact that the integration, the query evaluation and analysis happens on a system directly controlled by the user guarantees her privacy.

To illustrate the possibly very rich derivation of knowledge, suppose that a user, say Alice, met a friend, say Bob. The agenda indicates the time of a meeting with “Bob”. Some text messages may remove the ambiguity on which Bob she met (Alice may know several Bobs). The phone number he used has long been aligned with a specific Bob. The location of Alice, as provided by her phone GPS, provides the name of the bar where they met. Bob’ timeline in his social network provides pictures of the event. Finally, from her search history, we can even learn about topics they discussed. A meeting that had traces in different systems turned into a rich event in the KB.

Designing such a personal KB is not easy: Data of completely different nature has to be modeled in a uniform manner, pulled into the knowledge base, and integrated with other data. Different from related work in the area of information integration, we cannot rely on outside sources, we cannot expect redundancy, and we have a very limited and particular set of attributes.

We will mention such a system, the Thymeflow system that for instance aligns events based on time, e.g., a meeting in calendar and a GPS position, or on space, an address in contacts and a GPS position.

Acknowledgments. I will discuss works performed with colleagues that I would like to thank:

- Benjamin André, Cozy Cloud
- Daniel Kaplan, Fing
- Amélie Marian, Rutgers University
- Thomas Pellissier Tanon, ENS Lyon
- Pierre Senellart, ENS
- Fabian M. Suchanek, Telecom ParisTech

The bibliography provides references to these works. More references can of course be found in these papers.

References

1. Abiteboul, S., Andr, B., Kaplan, D.: Managing your digital life. *Commun. ACM* **58**(5), 32–35 (2015)
2. Abiteboul, S., Marian, A.: Personal information management systems. In: Tutorial, EDBT/ICDT Conference (2015). <https://www.slideshare.net/ameliemarian>
3. Montoya, D., Tanon, T.P., Abiteboul, S., Suchanek, F.M.: Thymeflow, a personal knowledge base with spatio-temporal data. In: CIKM 2016, pp. 2477–2480 (2015)
4. Montoya, D., Abiteboul, S., Senellart, P.: Hup-me: inferring and reconciling a timeline of user activity from rich smartphone data. In: SIGSPATIAL/GIS 2015, pp. 62:1–62:4 (2015)

Contents

Privacy on Personal Data Based Applications

Your Privacy, My Privacy? On Leakage Risk Assessment in Online Social Networks	3
<i>Ruggiero Gaetano Pensa and Livio Bioglio</i>	
From Self-data to Self-preferences: Towards Preference Elicitation in Personal Information Management Systems	10
<i>Tristan Allard, Tassadit Bouadi, Joris Duguépéroux, and Virginie Sans</i>	
Assessing Privacy Risk in Retail Data	17
<i>Roberto Pellungrini, Francesca Pratesi, and Luca Pappalardo</i>	
Differential Privacy and Neural Networks: A Preliminary Analysis	23
<i>Giuseppe Manco and Giuseppe Pirrò</i>	
Co-clustering for Differentially Private Synthetic Data Generation	36
<i>Tarek Benkhelif, Françoise Fessant, Fabrice Clérot, and Guillaume Raschia</i>	
Personal Analytics: An Individual and Collective Perspective	
Evaluating the Impact of Friends in Predicting User's Availability in Online Social Networks	51
<i>Andrea De Salve, Paolo Mori, and Laura Ricci</i>	
Movement Behaviour Recognition for Water Activities	64
<i>Mirco Nanni, Roberto Trasarti, and Fosca Giannotti</i>	
Automatic Recognition of Public Transport Trips from Mobile Device Sensor Data and Transport Infrastructure Information	76
<i>Mikko Rinne, Mehrdad Bagheri, Tuukka Tolvanen, and Jaakko Hollmén</i>	
Guess the Movie - Linking Facebook Pages to IMDb Movies.	98
<i>Paolo Fornacciari, Barbara Guidi, Monica Mordonini, Jacopo Orlandini, Laura Sani, and Michele Tomaiuolo</i>	

Research on Online Digital Cultures - Community Extraction and Analysis by Markov and k-Means Clustering.	110
<i>Giles Greenway, Tobias Blanke, Mark Cote, and Jennifer Pybus</i>	
Churn Prediction Using Dynamic RFM-Augmented Node2vec	122
<i>Sandra Mitrović, Bart Baesens, Wilfried Lemahieu, and Jochen De Weerdt</i>	
Multi-scale Community Detection in Temporal Networks Using Spectral Graph Wavelets	139
<i>Zhana Kuncheva and Giovanni Montana</i>	
Influence Maximization-Based Event Organization on Social Networks	155
<i>Cheng-Te Li</i>	
Author Index	159

Privacy on Personal Data Based Applications

Your Privacy, My Privacy? On Leakage Risk Assessment in Online Social Networks

Ruggero Gaetano Pensa^(✉) and Livio Bioglio

Department of Computer Science, University of Turin, Turin, Italy
{ruggero.pensa,livio.bioglio}@unito.it

Abstract. The problem of user privacy enforcement in online social networks (OSN) cannot be ignored and, in recent years, Facebook and other providers have improved considerably their privacy protection tools. However, in OSN's the most powerful data protection “weapons” are the users themselves. The behavior of an individual acting in an OSN highly depends on her level of privacy attitude, but, in this paper, we show that user privacy is also influenced by contextual properties (e.g., user's neighborhood attitude, the general behavior of user's subnetwork) and define a context-aware privacy score to measure a more realistic user privacy risk according to the network properties.

Keywords: Privacy metrics · Online social networks
Information spread

1 Introduction

The problem of user privacy in the so-called “Big Data Era” cannot be ignored and many companies are realizing the necessity to consider it at every stage of their business. In practice, they have been turning to the principle of *Privacy by Design* [2] by integrating privacy requirements into their business model. Online social network (OSN) providers are embracing this model as well. However, in OSN's the most powerful data protection “weapons” are the users themselves. The behavior of an individual in these situations highly depends on her level of privacy awareness: an aware user tends not to share her private information, or the private information of her friends, while an unaware user could not recognize some information as private, and could share it to her contacts, even to untrusted ones, putting her privacy or the privacy of her friends at risk. Users' privacy awareness then turns into the so-called “privacy attitude”, i.e., the users' willingness to disclose their own personal data to other users, that can be measured by leveraging users' privacy settings in social networking platforms [5].

A new question may arise now: how safe is the privacy of a social network user who is mostly surrounded by friends with a good privacy attitude? The question is not trivial, since the way most people set their privacy settings is based on the notion of closeness: close friends are usually allowed to see all user's updates, while acquaintances can only see “public” or less sensitive updates. The common

assumption is that closed friends are trusted ones and thus will not disclose friends' posts to other friends. In this paper, we model the effects of privacy attitude on information propagation in social networks with the aim of studying what happens to information diffused to friends with different levels of privacy awareness. The outcomes of this study lead to the intuition that privacy risk in a social network may be modeled similarly as page authority in a hyperlink graph of web pages. In fact, it is a well-known fact that more authoritative websites are likely to receive more links from other authoritative websites. Our hypothesis is that we may transpose the concept of "importance" of a web-page into the concept of "privacy risk" of users in a social network as follows: the more an individual is surrounded by friends that are careless about their privacy, the less the individual her/himself is likely to be protected from privacy leakage.

With the final goal of enhancing users' privacy awareness in online social networks, in this paper we propose a new context-aware privacy score based on personalized Pagerank [4], one of the most popular algorithms to rank web pages based on a personalized view of their importance (or authority). We show the effectiveness of our privacy measure on a large network of real Facebook users.

2 Information Diffusion vs. Privacy

We consider a social graph G composed by a set of n nodes $\{v_1, \dots, v_n\}$ representing the users of G . We represent the social network as a directed graph $G(V, E)$, where V is a set of n nodes and E is a set of directed edges $E = \{(v_i, v_j)\}$. Given a pair of nodes $v_i, v_j \in V$, $(v_i, v_j) \in E$ iff there exists a link from v_i to v_j (e.g., users v_i is in the friend list/circle of v_j or v_j follows v_i). We define the neighborhood of a node $v_i \in V$ as the set of nodes v_k directly connected to the node v_i , i.e., $\mathcal{N}(v_i) = \{v_k \in V \mid (v_i, v_k) \in E\}$. Finally, we consider a set P of privacy classes, representing the propensity of a user of the class to disclose her own or other's information. Each user of G belongs to a privacy class $p \in P$.

We employ an extension, proposed by us in [1], of the SIR model for considering the explicit or implicit privacy policies of an individual during the spread of information on a social network. A privacy class in the set $P = \{p_0, p_1, \dots, p_N\}$ is assigned to the susceptible (S) and infectious (I) compartments, representing the privacy attitude of an individual belonging to the compartment, and consequently her behavior on information spreading, from less aware (p_0) to more aware (p_N). This behavior is reached by assigning different values to parameters λ (infection probability) and μ (recovery probability) of each privacy class: every privacy class $p \in P$ is linked to a different pair of values λ_p and μ_p . We also introduce a novel parameter $\beta_p \in [0, 1]$ in the SIR model, symbolizing the interest in information of the users in privacy class p . We denote with $c(v_i, t) \in \{S, I, R\}$ the compartment of user v_i at time t . The evolution probabilities are obtained as follows. Let $p(v_i) = p \in P$ be the privacy class of a user v_i . If it belongs to the susceptible compartment, it may be infected at time $t + 1$ with probability: $P_{inf}(v_i, t + 1) = \beta_p \cdot \left(1 - \prod_{p' \in P} (1 - \lambda_{p'})^{n_I(v_j, t)}\right)$, where $n_I(v_j, t) = |\{v_j \in \mathcal{N}(v_i) \mid c(v_j, t) = I \wedge p(v_j) = p'\}|$ is the number of

individuals in the I compartment and privacy class p' at time t among the neighbors of individual v_i . Otherwise, if the individual v_i belongs to the infectious (I) compartment at time t , it may recover with probability μ_p at time $t + 1$.

We now provide the results of our experiments performed over a Facebook-like synthetic network generated using LDBC – SNB Data Generator which produces graphs that mimic the characteristics of real Facebook networks [3]: in particular, we generate a network with 80,000 nodes, but here we consider only the greatest connected component of such network, composed by approximately 75,000 nodes and 2,700,000 edges. See [1] for the details of our experiments.

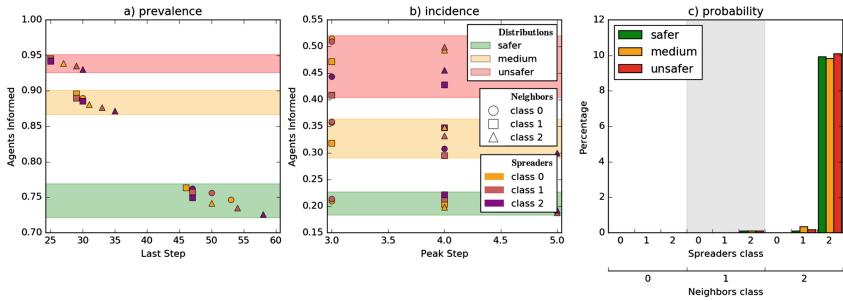


Fig. 1. (a) Median of prevalence of informed individuals (ratio) in each set of simulations. (b) Median of incidence of informed individuals (ratio) in each set of simulations. (c) Probability that information does not reach more than 1% of the population. (Color figure online)

The features extracted from our simulations are graphically summarized in Fig. 1(a) for prevalence curves and in Fig. 1(b) for incidence curves. For each point of these graphs, the marker color identifies the privacy class of the initial spreader, its shape identifies the privacy class of the neighbors of initial spreader node, while its background color identifies the distribution of privacy classes of the nodes in the whole network. Each point shows the median value resulting from 100 simulations performed on 10 initial spreaders.

The most noticeable result is the role of the attitude towards privacy of the initial spreader and its neighbors: a safer attitude of the node and its neighbors decreases the portion of informed population, and extends the duration of information diffusion, but its impact is not as influential as the behavior of the whole network. For an aware user, even if the probability of diffusing information to a friend is low, the number of friends is so high that a small number of friends can become spreaders themselves. As soon as information reaches a node out of the neighborhood, its diffusion depends only on the attitude on privacy of the whole network. For this reason we decide to analyze the portion of simulations where information has reached only a small portion of the population, lower than 1%, before dying: our results are reported in Fig. 1(c). In this case we notice that the attitude of the network is irrelevant, and only the privacy classes of the spreader

and its neighborhood is crucial. Interestingly, a safe attitude of the spreader or of the neighbors is not sufficient on its own to block information diffusion. An information item on a safer user with unsafer friends, and vice versa, can easily overtake the circle of friends.

These observations lead to the following crucial consideration: to assess the objective privacy risk of users, one cannot simply take into account their propensity to self-protection. The attitude of their neighbors also counts, as well as the attitude of the whole subnetwork in which their interact. In the next section, we will present a measure that quantifies the privacy leakage of users considering the risks due not only to their attitude towards privacy but also to the attitude of their friends and subnetwork.

3 A Context-Aware Privacy Metrics

We consider the social graph $G(V, E)$ defined in Sect. 2 and associate, to each user $v_i \in V$, an *intrinsic privacy risk* $\rho_p(u_i)$, which is defined as the user propensity to privacy leakage. The assumption is that some users are more prone to disclose their personal data than others and, thus, they are intrinsically more at risk. In the following, we instantiate $\rho_p(u_i)$ according to the *P-Score* proposed by Liu and Terzi [5], an established and reliable definition of privacy score.

As shown in Sect. 2, if a user is mostly surrounded by friends (or friends of friends) that do not care that much about privacy, then she should be more exposed to privacy leakage than a user who is principally connected to friends that care about their own (and others') privacy. This consideration leads to the intuition that privacy risk in a social network may be modeled similarly as page authority in a hyperlink graph of web pages. Hence, we transpose the concept of “importance” of a web-page into the concept of “privacy risk” of users in a social network as follows: the more an individual is surrounded by friends that are careless about their privacy, the less the individual her/himself is likely to be protected from privacy leakage. Hence, we correct the *P-Score* by using *personalized Pagerank* [4], one of the most popular algorithms to rank web pages based on their centrality (or authority). It is used to create a personalized view of the relative importance of the nodes. We can now introduce our context-aware privacy score (called *CAP-Score*), defined by the following distribution:

$$\mathbf{P}^\rho = d\mathbf{A}^\top \mathbf{P}^\rho + \frac{(1-d)}{\sum_{k=1}^n \rho_p(u_k)} \boldsymbol{\rho} \quad (1)$$

where $\boldsymbol{\rho} = [\rho_p(u_1), \dots, \rho_p(u_n)]^\top$, $\mathbf{P}^\rho = [p^\rho(v_1), \dots, p^\rho(v_n)]^\top$ is the *CAP-Score* vector ($p^\rho(v_i)$ being the *CAP-Score* associated to vertex v_i), $d = [0, 1]$ is the damping factor (the $1 - d$ quantity is also known as restart probability) and \mathbf{A} is a $n \times n$ matrix such that each element $a_{ij} = 1/\deg^+(v_i)$ ($\deg^+(v_i)$ being the outdegree of v_i) if $(v_i, v_j) \in E$ ($a_{ij} = 0$ otherwise).

An example of context-aware score computation is given in Fig. 2. Figure 2(a) is an example of graph where an aware user (the central one) is surrounded by

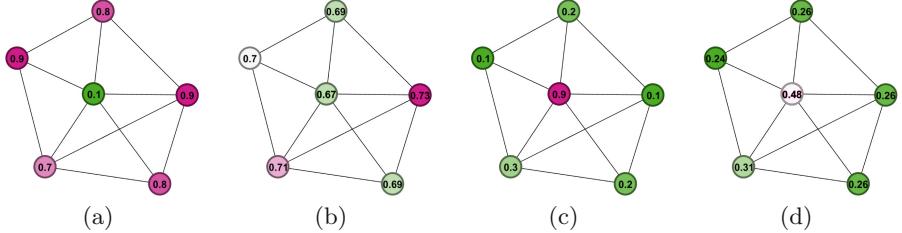


Fig. 2. Context-aware scores (b and d) in two differently aware networks (a and c).

users with high intrinsic risk. Figure 2(b) represents the same network with the computed *CAP-Scores*: the score value of the central user is adjusted according to the context and it is higher than in Fig. 2(a). Instead, Fig. 2(c) shows a network with the same topology but different intrinsic risks. In particular the unaware central user (with high risk) is surrounded by rather aware users (with low privacy risk). In this case, our measure for the central user is revised upwards (see Fig. 2(d)), according to a context in which all other users form a kind of barrier protecting the privacy of the central users.

4 Experimental Results

In this section, we show experimentally the improved effectiveness of our *CAP-Score* w.r.t. information diffusion phenomena. In our experiments we use a snapshot of the Facebook graph consisting on the ego-networks of real Facebook users gathered leveraging an online experiment described in [6]. It is a graph with 75,193 nodes and 1,377,672 edges, with the largest connected component consisting of 73,050 nodes and 1,333,276 edges. The degree distribution of the network is given in Fig. 3(a). For 101 users, who replied to a specific survey (specified in [6] as well), we have computed the *P-Score* [5].

We study the relationship between the two definitions of privacy score (*P-Score* and *CAP-Score*) and the effects of information propagation across the network. A good privacy score should take into account the number of nodes that may potentially access and diffuse some information coming from other nodes in the same network. For this reason, we perform several Monte Carlo simulations of an information propagation scenario within our snapshot of Facebook. In particular, we adopted the Susceptible-Infectious-Recovered (SIR) epidemic model. In our experiments, for all nodes we set an infection probability $\lambda = 0.5$ and a recovery probability $\mu = 0.3$. Then, we select N seed nodes that, in turn, are considered as the individuals that start the infection (i.e., information diffusion process) and measure the number of nodes (*prevalence rate*) that are either infected (I) or recovered (R) after each step of the simulation. The seed nodes are the 101 Facebook users for which we have measured the *P-Score*. Finally, for each simulation step we compute the Spearman's ρ coefficient between the prevalence rate and the two privacy scores. The results are reported in Fig. 3.

Interestingly, the gap between the *P-Score*'s ρ and our context-aware score's ρ in the very first iterations is significantly large. Assuming that the *P-Score* correctly measures the privacy risk based on users' privacy preferences, a possible explanation is that the users underestimate their centrality within the network. Although our Facebook snapshot cannot be considered a statistically valid sample of the entire Facebook graph, the huge difference in terms of correlation with the prevalence rate confirms that privacy leakage metrics should not ignore the context in which the users operate within the social network.

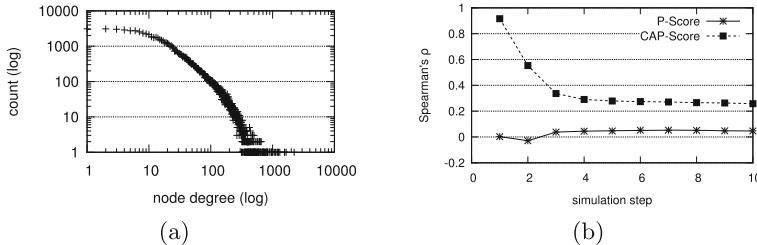


Fig. 3. Degree distribution (a) and correlation between prevalence and privacy (b).

5 Conclusions

In this paper, we have shown how privacy attitude can affect the diffusion of information on social networks and, with the final goal of supporting users' privacy awareness in online social networks, we have proposed a context-aware definition of privacy score. This measure, as shown in our experiments, is a good estimate of the objective privacy risk of the users. Based on these results, we believe that our framework can be easily plugged into any domain-specific or general-purpose social networking platforms, thus inspiring the design of privacy-preserving social networking components for *Privacy by Design* compliant software [2].

Acknowledgments. This work was supported by Fondazione CRT (grant number 2015-1638).

References

1. Bioglio, L., Pensa, R.G.: Impact of neighbors on the privacy of individuals in online social networks. In: Proceedings of ICCS 2017, pp. 28–37 (2017)
2. Cavoukian, A.: Privacy by design [leading edge]. IEEE Technol. Soc. Mag. **31**(4), 18–19 (2012)
3. Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat-Pérez, A., Pham, M., Boncz, P.A.: The LDBC social network benchmark: interactive workload. In: Proceedings of ACM SIGMOD 2015, pp. 619–630. ACM (2015)

4. Jeh, G., Widom, J.: Scaling personalized web search. In: Proceedings of WWW 2003, pp. 271–279. ACM (2003)
5. Liu, K., Terzi, E.: A framework for computing the privacy scores of users in online social networks. TKDD **5**(1), 6:1–6:30 (2010)
6. Pensa, R.G., Di Blasi, G.: A privacy self-assessment framework for online social networks. Expert Syst. Appl. **86**, 18–31 (2017)

From Self-data to Self-preferences: Towards Preference Elicitation in Personal Information Management Systems

Tristan Allard^{1(✉)}, Tassadit Bouadi^{1(✉)}, Joris Duguéperoux^{1,2},
and Virginie Sans¹

¹ Univ. Rennes 1/IRISA, Rennes, France

{tristan.allard,tassadit.bouadi,joris.dugueperoux,
virginie.sans}@irisa.fr

² ENS Rennes, Rennes, France

joris.dugueperoux@ens-rennes.fr

Abstract. Ever-increasing quantities of personal data are generated by individuals, knowingly or unconsciously, actively or passively (*e.g.*, bank transactions, geolocations, posts on web forums, physiological measures captured by wearable sensors). Most of the time, this wealth of information is stored, managed, and valorized in isolated systems owned by private companies or organizations. Personal information management systems (PIMS) propose a groundbreaking counterpoint to this trend. They essentially aim at providing to any interested individual the technical means to re-collect, manage, integrate, and valorize his/her own data through a dedicated system that he/she owns and controls. In this vision paper, we consider personal preferences as first-class citizens data structures. We define and motivate the *threefold preference elicitation problem in PIMS* - elicitation from local personal data, elicitation from group preferences, and elicitation from user interactions. We also identify hard and diverse challenges to tackle (*e.g.*, small data, context acquisition, small-scale recommendation, low computing resources, data privacy) and propose promising research directions. Overall, we hope that this paper uncovers an exciting and fruitful research track.

Keywords: Preferences · Privacy · Self-data
Personal information management system

1 Introduction

An ever-increasing quantity and diversity of personal data feeds the database systems of various companies (*e.g.*, emails, shopping baskets, news, geolocations, physiological measures, electrical consumption, movies, social networks, posts on forums, professional resumes). Although individuals often benefit indirectly from this large-scale systematic capture of their data (*e.g.*, free access to services), the use value they get from it remains strongly limited by its fragmentation

in non-cooperative *data silos* and by the usage allowed and supported by each silo [1,5]. Personal information management systems (*PIMS* for short) aim at giving to individuals technical means to re-collect, integrate, manage, and use their data (or at least a part of it) in a central location *under their control* (e.g., a personal computer, a rented virtual machine). The expected uses of a *PIMS* are those of a full-fledged data-centric personal assistant, including for example panoramic integrated personal data visualizations (e.g., health data from health centres and physiological measures from wearables), vendor relationship management and privacy-preserving recommendations (e.g., comparing offers from electricity providers given one's detailed electrical consumption), automatic completion of online profiles (e.g., privacy preferences on a social network)¹.

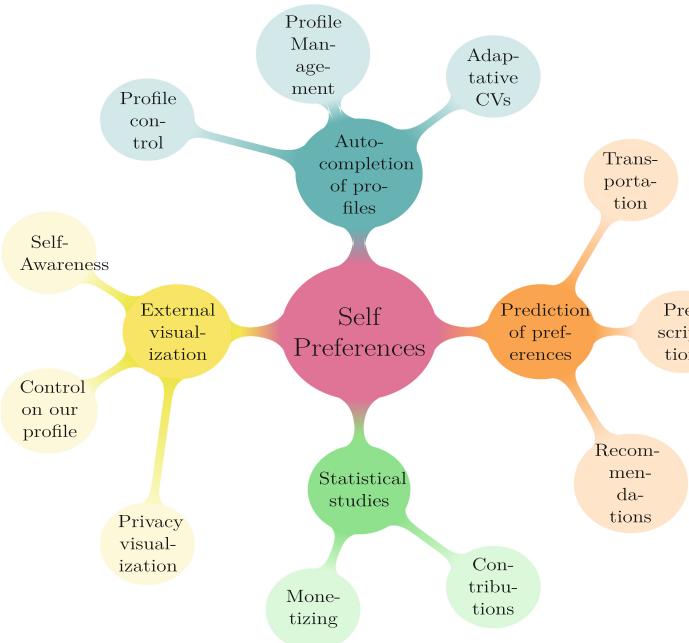


Fig. 1. A non-exhaustive list of uses of self-preferences

We are moreover experiencing today a strong push towards the widespread adoption of *PIMS*: various commercial industrial-strength *PIMS* exist today (e.g., Cozy², Hub of All Things³), modern laws in favor of data portability are passed (e.g., article 20 of the new European General Data Protection Regulation⁴, article 48 of the new French data protection bill⁵), institutions

¹ See, e.g., <https://tinyurl.com/mesinfosValue> for a large variety of use-cases.

² <https://cozy.io/en/>.

³ <http://hubofallthings.com/>.

⁴ <https://tinyurl.com/euDataPort>.

⁵ <https://tinyurl.com/frDataPort>.

launch initiatives and express opinions in favor of PIMS (*e.g.*, the European Data Protection Supervisor opinion about PIMS⁶, the US MyData initiatives⁷, the UK MiData initiative⁸), and the research field is active (*e.g.*, [1, 2, 5, 6]).

However the promises that PIMS carry on will be strongly hampered if they fail in eliciting *personal preferences* from the wealth of personal data they store, simply because they crucially rely on accurately modeling, reasoning, and using personal preferences. Consider for example an automatic profile completion application based on preferences (see Fig. 1 for other examples of uses). It is often time-consuming and error-prone to fill in the forms about, *e.g.*, privacy preferences. A detailed automatic completion application based on preferences would for example precisely tune the privacy preferences of social networks depending on their application domains (*e.g.*, a friendship-centric network would be parameterized differently than a professional social network).

In this vision paper, we advocate for considering personal preferences as first-class citizens in PIMS. We call them *self-preferences*, characterize the PIMS computing environment, and identify key challenges to be solved for eliciting self-preferences from the personal data stored in a PIMS. These challenges fall in three categories: local elicitation (based on the data that is stored locally), global elicitation (or group elicitation, based on the data of similar individuals), and interactive elicitation (based on the individual's feedback). They involve various fields of research such as, *e.g.*, preference elicitation, small data, data integration, data privacy, distributed computing, data visualization.

This paper is organized as follows. Section 2 defines more precisely the problem of self-preferences elicitation in PIMS. Section 3 identifies the main related key challenges to be tackled. Finally, Sect. 4 concludes.

2 Problem Statement

2.1 Personal Information Management Systems

A *Personal Information Management Systems* (or *PIMS* for short) is basically a suite of software in charge of collecting, centralizing, and managing the personal data related to an individual. In this work, we focus on PIMS that are hosted and run on a server *controlled by the individual* [1] (*e.g.*, owned or rented). A PIMS may be executed in a large variety of settings: a personal device possessed by the individual (*i.e.*, self-hosting model - Cozy for example), a virtual machine rent by the individual to a cloud provider (*i.e.*, platform-as-a-service model - Hub of all things for example). Despite such heterogeneity, we believe that most PIMS can be characterized as follows:

Resources. Whatever the PIMS execution environment, it is dedicated to serve a single individual, or at most a few closely related individuals (*e.g.*, the members of a family). Therefore, the local computing resources—CPU and RAM—are

⁶ <https://tinyurl.com/edpsOnPims>.

⁷ <https://tinyurl.com/usMyData>.

⁸ <https://tinyurl.com/ukMiData>.

scarce. Typical worst-case resources are on the order of those provided by a low-cost personal device (*e.g.*, a raspberry-pi—around 1 GB RAM and 1.2 GHz CPU), although on average they are probably similar to current commodity hardware. More resources may be available when the execution environment is a virtual machine in the cloud, but they remain limited to the personal space of the individual.

Threat Models. First, a PIMS is trusted by its owner. It is actually a pre-requirement because it stores the personal data together with the credentials required for fetching data from the web (couples of login/password). Second, a PIMS is not necessarily trusted by other individuals. For example, it may be hosted by a controversial cloud provider, or self-hosted and possibly ill-protected. When several PIMS collaborate for performing a global computation (*e.g.*, statistics over a population), the personal data involved in the computation need to be protected from dubious PIMS. Typical attack models include the *honest-but-curious* model (observes any information that leaks during the protocol) and the *malicious* model (may additionally deviate from the protocol - *e.g.*, by forging or tampering messages). Resistance to *collusions* of PIMS must also be considered.

2.2 Preference Model

Preferences express comparisons on a set X of items, choices or alternatives. Preferences could be expressed in different forms. We can express preferences in a quantitative way, by expressing degrees of interest in terms of scores (*e.g.*, “*My preference for red wine is 0.5 and for white wine is 0.3*”), or in a qualitative way, by pairwise comparisons or other AI preference formalisms (*e.g.*, “*I prefer red wine to white wine*”). The qualitative approach is more general than the quantitative one.

More formally, a preference relation (*i.e.*, *preference order*) $\succeq \subseteq X \times X$ is a reflexive, antisymmetric and transitive binary relation, where $x \succeq y$ means “*x is at least as good as y*”. Different types of preference orders can be defined, depending on the properties they satisfy (symmetry, transitivity, etc.).

Furthermore, preferences are of various kinds and take different forms (*e.g.*, *fuzzy preferences*, *conditional preferences*, *bipolar preferences*, *etc.*). Several preference modeling formalisms have been proposed in the literature, each of which allows to express and model some properties of preferences. Thus, by comparing the expressive power of these formalisms, one can choose the adequate formalism that accurately describes and captures the rich cognitive model of an individual.

In real-life applications, preferences fully depend on the decision context. We believe that context-aware models such as, *e.g.*, the conditional preference network formalism (CP-net) [4] is especially adequate for self-preferences. This formalism provides an intuitive way to specify conditional or contextual statements of the form “*If context or condition C is true, then preference P is true*” (*e.g.*, “*If the main course is meat, I prefer red wine to white wine*”).

2.3 Elicitation of Self-preferences: A Threefold Problem

The general purpose of this work is to elicit and maintain the self-preferences of an individual, in a PIMS context, for letting him/her use, manage, and visualize them. The problem is actually threefold, depending on the actual source of information used for the self-preference elicitation (we do not pretend to have listed all the potential information sources for the self-preference elicitation problem).

Problem 1: Local Elicitation. The primary source of information is the personal data stored in the PIMS. Problem 1 is thus the local elicitation of self-preferences. It can be phrased as follows: *which model(s) would be adequate for self-preferences and how to elicit them on commodity hardware based on heterogeneous and small data?*

Problem 2: Global Elicitation. The secondary source of information is the self-preferences of groups of *similar* individuals. Indeed, PIMS are usually connected to the Internet (even if they may get disconnected arbitrarily) and can participate in distributed algorithms provided that the personal data and preferences they store remain adequately protected. The preferences of similar individuals may thus enrich local self-preferences. Problem 2 is thus the global, group-based, elicitation of self-preferences: *How to form meaningful group preferences shared by a significant number of individuals—in a privacy-preserving manner and from commodity hardware devices—and enrich local self-preferences accordingly?*

Problem 3: Manual Elicitation. Finally, the third available source of information is the individual him/her-self. Moreover, in a real-life context, self-preferences are not static (*e.g.*, change of beliefs, new self-preferences learned, new data sources feeding the PIMS). As a result, problem 3 is the refinement and revision by the individual of the self-preferences already stored in his/her PIMS (*e.g.*, elicited locally or globally). *How to guide a non-expert individual for letting him/her interactively refine or revise his/her self-preferences and perform the updates efficiently on commodity hardware?*

3 Challenges

We identify below key challenges to be addressed for solving each of the three problems stated in Sect. 2.

Challenge 1: Local is Small. The idea of accurately reflecting and expressing individuals' preferences is a common and known AI issue (see, *e.g.*, preference mining [8] and preference learning [7] techniques). Consequently, previous works have coped with this problem, leading to a sophistication of the preference models and thus to more tedious and resource-consuming elicitation methods. Proposing effective preference elicitation techniques becomes more and more challenging when facing *small and heterogeneous data* and having *low computing resources*. First, we have to *integrate and reason* about data from different sources, and this in a consistent manner. We think that providing the system

with *context metadata* will ease the integration process. Second, the *small scale of local data* is a challenge to overcome for accurately learning even the simplest models. Lastly, we have to *develop optimal and efficient local elicitation algorithms* to address the low computing resources of a typical PIMS.

Challenge 2: Global is Threatening. In order to benefit from the self-preferences of similar individuals for enriching the local self-preferences, three steps are necessary: (1) groups of similar individuals must be formed, (2) global group preferences must be aggregated from local self-preferences, and (3) *some* group preferences must be selected for enriching local self-preferences. Performing these steps (possibly combined together) in a distributed and privacy-preserving manner is a difficult challenge. Several previous works have already tackled problems similar to step 1 (*e.g.*, privacy-preserving distributed clustering, k -nearest neighbors)—although usually the underlying data is not preferences or the underlying model is not as sophisticated as elaborate preference models. We believe that the main difficulties lie in step 2 and step 3. Indeed, aggregating together a set of preferences (see, *e.g.*, [3]) may require to perform operations not supported efficiently by encryption schemes (*e.g.*, comparison operators for implementing MIN aggregate, or threshold computations). Moreover, the low computing resources of PIMS may not be able to cope with encryption-intensive algorithms, calling for alternative protection-by-perturbation strategies (*e.g.*, differential privacy) and consequently the design of specific perturbation mechanisms for preferences and specific privacy/utility tradeoffs. Finally, the challenge gets even harder when considering that preference models may be heterogeneous across PIMS.

Challenge 3: Manual is Tedious. With the possibility of interactive revision and refinement, many challenges arise. Revising and refining contextual preferences may require to merge (1) preferences of the same context, (2) preferences with context hierarchically associated, (3) preferences with different contexts having similar ontologies. This leads to design robust techniques for the acquisition of the proposed preference model. Furthermore, these techniques should fit limited-resource systems. The interactivity aspect will also benefit from comprehensive preference-human interfaces, that should provide mechanisms to express rich preference models while keeping user effort to an acceptable level.

4 Conclusion

In this vision paper, we discuss the value of self-preferences for their owners, foreshadow an illustrative but not exhaustive list of the potential self-preferences uses, define the *threefold self-preference elicitation problem in PIMS*, and identify key research challenges that need to be tackled in order to help individuals to express, use, and manage their self-preferences. We hope this paper highlights promising research avenues in the field of self-data management.

References

1. Abiteboul, S., André, B., Kaplan, D.: Managing your digital life. *Commun. ACM* **58**(5), 32–35 (2015)
2. Abiteboul, S., Arenas, M., Barceló, P., Bienvenu, M., Calvanese, D., David, C., Hull, R., Hüllermeier, E., Kimelfeld, B., Libkin, L., Martens, W., Milo, T., Murlak, F., Neven, F., Ortiz, M., Schwentick, T., Stoyanovich, J., Su, J., Suciu, D., Vianu, V., Yi, K.: Research directions for principles of data management (abridged). *SIGMOD Rec.* **45**(4), 5–17 (2017)
3. Basu Roy, S., Lakshmanan, L.V., Liu, R.: From group recommendations to group formation. In: Proceedings of SIGMOD 2015, pp. 1603–1616 (2015)
4. Boutilier, C., Brafman, R.I., Domshlak, C., Hoos, H.H., Poole, D.: CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res.* **21**, 135–191 (2004)
5. de Montjoye, Y., Wang, S.S., Pentland, A.: On the trusted use of large-scale personal data. *IEEE Data Eng. Bull.* **35**(4), 5–8 (2012)
6. Estrin, D.: Small data, where $N = Me$. *Commun. ACM* **57**(4), 32–34 (2014)
7. Fürnkranz, J., Hüllermeier, E.: Preference learning: an introduction. In: Fürnkranz, J., Hüllermeier, E. (eds.) *Preference learning*, pp. 1–17. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14125-6_1
8. Holland, S., Ester, M., Kießling, W.: Preference mining: a novel approach on mining user preferences for personalized applications. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) *PKDD 2003. LNCS (LNAI)*, vol. 2838, pp. 204–216. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39804-2_20

Assessing Privacy Risk in Retail Data

Roberto Pellungrini^{1(✉)}, Francesca Pratesi^{1,2}, and Luca Pappalardo^{1,2}

¹ Department of Computer Science, University of Pisa, Pisa, Italy

roberto.pellungrini@di.unipi.it

² ISTI-CNR, Pisa, Italy

Abstract. Retail data are one of the most requested commodities by commercial companies. Unfortunately, from this data it is possible to retrieve highly sensitive information about individuals. Thus, there exists the need for accurate individual privacy risk evaluation. In this paper, we propose a methodology for assessing privacy risk in retail data. We define the data formats for representing retail data, the privacy framework for calculating privacy risk and some possible privacy attacks for this kind of data. We perform experiments in a real-world retail dataset, and show the distribution of privacy risk for the various attacks.

1 Introduction

Retail data are a fundamental tool for commercial companies, as they can rely on data analysis to maximize their profit [7] and take care of their customers by designing proper recommendation systems [11]. Unfortunately, retail data are also very sensitive since a malicious third party might use them to violate an individual's privacy and infer personal information. An adversary can re-identify an individual from a portion of data and discover her complete purchase history, potentially revealing sensitive information about the subject. For example, if an individual buys only fat meat and precooked meal, an adversary may infer a risk to suffer from cardiovascular disease [4]. In order to prevent these issues, researchers have developed privacy preserving methodologies, in particular to extract association rules from retail data [3,5,10]. At the same time, frameworks for the management and the evaluation of privacy risk have been developed for various types of data [1,2,8,9,13].

We propose privacy risk assessment framework for retail data which is based on our previous work on human mobility data [9]. We first introduce a set of data structures to represent retail data and then present two re-identification attacks based on these data structures. Finally, we simulate these attacks on a real-world retail dataset. The simulation of re-identification attacks allows the data owner to identify individuals with the highest privacy risk and select suitable privacy preserving technique to mitigate the risk, such as k -anonymity [12].

The rest of the paper is organized as follows. In Sect. 2, we present the data structures which describe retail data. In Sect. 3, we define the privacy risk and the re-identification attacks. Section 4, shows the results of our experiments and, finally, Sect. 5 concludes the paper proposing some possible future works.

2 Data Definitions

Retail data are generally collected by retail companies in an automatic way: customers enlist in membership programs and, by means of a loyalty card, share informations about their purchases while at the same time receiving special offers and bonus gifts. Products purchased by customers are grouped into baskets. A basket contains all the goods purchased by a customer in a single shopping session.

Definition 1 (Shopping Basket). *A shopping basket S_j^u of an individual u is a list of products $S_j^u = \{i_1, i_2, \dots, i_n\}$, where i_h ($h = 1, \dots, n$) is an item purchased by u during her j -th purchase.*

The sequence of an individual's baskets forms her shopping history related to a certain period of observation.

Definition 2 (History of Shopping Baskets). *The history of shopping baskets HS^u of an individual u is a time-ordered sequence of shopping baskets $HS^u = \{S_1^u, \dots, S_m^u\}$.*

3 Privacy Risk Assessment Model

In this paper we start from the framework proposed in [9] and extended in [8], which allows for the assessment of the privacy risk in human mobility data. The framework requires the identification of the minimum data structure, the definition of a set of possible attacks that a malicious adversary might conduct on an individual, and the simulation of these attacks. An individual's privacy risk is related to her probability of re-identification in a dataset w.r.t. a set of re-identification attacks. The attacks assume that an adversary gets access to a retail dataset, then, using some previously obtained background knowledge, i.e., the knowledge of a portion of an individual's retail data, the adversary tries to re-identify all the records in the dataset regarding that individual. We use the definition of privacy risk (or re-identification risk) introduced in [12].

The background knowledge represents how the adversary tries to re-identify the individual in the dataset. It can be expressed as a hierarchy of categories, configurations and instances: there can be many background knowledge categories, each category may have several background knowledge configurations, each configuration may have many instances. A background knowledge category is an information known by the adversary about a specific set of dimensions of an individual's retail data. Typical dimensions in retail data are the items, their frequency of purchase, the time of purchase, etc. Examples of background knowledge categories are a subset of the items purchased by an individual, or a subset of items purchased with additional spatio-temporal information about the shopping session. The number k of the elements of a category known by the adversary gives the background knowledge configuration. This represents the fact that the quantity of information that an adversary has may vary in size. An

example is the knowledge of $k = 3$ items purchased by an individual. Finally, an instance of background knowledge is the specific information known, e.g., for $k = 3$ an instance could be eggs, milk and flour bought together. We formalize these concepts as follows.

Definition 3 (Background knowledge configuration). *Given a background knowledge category \mathcal{B} , we denote by $B_k \in \mathcal{B} = \{B_1, B_2, \dots, B_n\}$ a specific background knowledge configuration, where k represents the number of elements in \mathcal{B} known by the adversary. We define an element $b \in B_k$ as an instance of background knowledge configuration.*

Let \mathcal{D} be a database, D a retail dataset extracted from \mathcal{D} (e.g., a data structure as defined in Sect. 2), and D_u the set of records representing individual u in D , we define the probability of re-identification as follows:

Definition 4 (Probability of re-identification). *The probability of re-identification $PR_D(d = u|b)$ of an individual u in a retail dataset D is the probability to associate a record $d \in D$ with an individual u , given an instance of background knowledge configuration $b \in B_k$.*

If we denote by $M(D, b)$ the records in the dataset D compatible with the instance b , then since each individual is represented by a single History of Shopping Baskets, we can write the probability of re-identification of u in D as $PR_D(d = u|b) = \frac{1}{|M(D, b)|}$. Each attack has a matching function that indicates whether or not a record is compatible with a specific instance of background knowledge.

Note that $PR_D(d = u|b) = 0$ if the individual u is not represented in D . Since each instance $b \in B_k$ has its own probability of re-identification, we define the risk of re-identification of an individual as the maximum probability of re-identification over the set of instances of a background knowledge configuration:

Definition 5 (Risk of re-identification or Privacy risk). *The risk of re-identification (or privacy risk) of an individual u given a background knowledge configuration B_k is her maximum probability of re-identification $Risk(u, D) = \max PR_D(d = u|b)$ for $b \in B_k$. The risk of re-identification has the lower bound $\frac{|D_u|}{|D|}$ (a random choice in D), and $Risk(u, D) = 0$ if $u \notin D$.*

3.1 Privacy Attacks on Retail Data

The attacks we consider in this paper consist of accessing the released data in the format of Definition (2) and identifying all users compatible with the background knowledge of the adversary.

Intra-basket Background Knowledge. We assume that the adversary has as background knowledge a subset of products bought by her target in a certain shopping session. For example, the adversary once saw the subject at the workplace with some highly perishable food, that are likely bought together.

Definition 6 (Intra-basket Attack). Let k be the number of products of an individual w known by the adversary. An Intra-Basket background knowledge instance is $b = S'_i \in B_k$ and it is composed by a subset of purchase $S'_i \subseteq S_j^w$ of length k . The Intra-Basket background knowledge configuration based on k products is defined as $B_k = S^{w[k]}$. Here $S^{w[k]}$ denotes the set of all the possible k -combinations of the products in each shopping basket of the history.

Since each instance $b = S'_i \in B_k$ is composed of a subset of purchase $S'_i \subseteq S_j^w$ of length k , given a record $d = HS^u \in D$ and the corresponding individual u , we define the matching function as:

$$\text{matching}(d, b) = \begin{cases} \text{true} & \exists S_j^d \mid S'_i \subseteq S_j^d \\ \text{false} & \text{otherwise} \end{cases} \quad (1)$$

Full Basket Background Knowledge. We suppose that the adversary knows the contents of a shopping basket of her target. For example, the adversary once gained access to a shopping receipt of her target. Note that in this case it is not necessary to establish k , i.e., the background knowledge configuration has a fixed length, given by the number of items of a specific shopping basket.

Definition 7 (Full Basket Attack). A Full Basket background knowledge instance is $b = S_j^w \in B$ and it is composed of a shopping basket of the target w in all her history. The Full Basket background knowledge configuration is defined as $B = S_i^w \in HS^w$.

Since each instance $b = S_i^w \in B$ is composed of a shopping basket S_i^w , given a record $d = HS^u \in D$ and the corresponding individual u , we define the matching function as:

$$\text{matching}(d, b) = \begin{cases} \text{true} & \exists S_j^d \mid S_i^w = S_j^d \\ \text{false} & \text{otherwise} \end{cases} \quad (2)$$

4 Experiments

For the Intra-basket attack we consider two sets of background knowledge configuration B_k with $k = 2, 3$, while for the Full Basket attack we have just one possible background knowledge configuration, where the adversary knows an entire basket of an individual. We use a retail dataset provided by Unicoop¹ storing the purchases of 1000 individuals in the city of Leghorn during 2013, corresponding to 659,761 items and 61,325 baskets. We consider each item at the category level, representing a more general description of a specific item, e.g., “Coop-brand Vanilla Yogurt” belongs to category “Yogurt”.

We performed a simulation of the attacks for all B_k . We show in Fig. 1 the cumulative distributions of privacy risks. For the Intra-basket attack, with $k = 2$ we have almost 75% of customers for which privacy risk is to equal 1.

¹ <https://www.unicooptirreno.it/>.

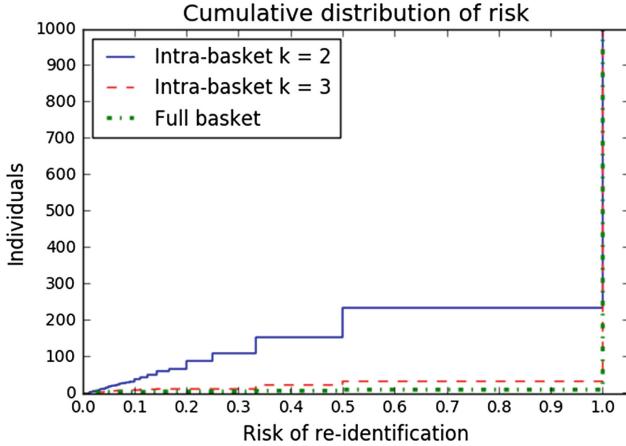


Fig. 1. Cumulative distributions for privacy attacks.

Switching to $k = 3$ causes a sharp increase in the overall risk: more than 98% of individuals have maximum privacy risk (e.g., 1). The difference between the two configurations is remarkable, showing how effective an attack could be with just 3 items. Since most of customers are already re-identified, further increasing the quantity of knowledge (e.g., exploiting higher k or the Full Basket attack) does not offer additional gain. Similar results were obtained for movie rating dataset in [6] and mobility data in [9], suggesting the existence of a possible general pattern in the behavior of privacy risk.

5 Conclusion

In this paper we proposed a framework to assess privacy risk in retail data. We explored a set of re-identification attacks conducted on retail data structures, analyzing empirical privacy risk of a real-world dataset. We found, on average, a high privacy risk across the considered attacks. Our approach can be extended in several directions. First, we can expand the repertoire of attacks by extending the data structures, i.e., distinguishing among shopping sessions and obtaining a proper transaction dataset, or considering different dimensions for retail data, e.g., integrating spatio-temporal informations about the purchases. Second, it would be interesting to compare the distributions of privacy risk of different attacks through some similarity measures, such as the Kolmogorov-Smirnov test. A more general and thorough approach to privacy risk estimation can be found in [14] and it would be interesting to extend our framework with its approaches. Another possible development is to compute a set of measures commonly used in retail data analysis and investigate how they relate to privacy risk. Finally, it would be interesting to generalize the privacy risk computation framework to data of different kinds, from retail to mobility and social media data, studying sparse relation spaces across different domains.

Acknowledgment. Funded by the European project SoBigData (Grant Agreement 654024).

References

1. Alberts, C., Behrens, S., Pethia, R., Wilson, W.: Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Framework, Version 1.0. CMU/SEI-99-TR-017. Software Engineering Institute, Carnegie Mellon University (1999)
2. Deng, M., Wuyts, K., Scandariato, R., Preneel, B., Joosen, W.: A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requir. Eng.* **16**, 1 (2011)
3. Giannotti, F., Lakshmanan, L.V., Monreale, A., Pedreschi, D., Wang, H.: Privacy-preserving mining of association rules from outsourced transaction databases. *IEEE Syst. J.* **7**(3), 385–395 (2013)
4. Kant, A.K.: Dietary patterns and health outcomes. *J. Am. Dietetic Assoc.* **104**(4), 615–635 (2004)
5. Le, H.Q., Arch-Int, S., Nguyen, H.X., Arch-Int, N.: Association rule hiding in risk management for retail supply chain collaboration. *Comput. Ind.* **64**(7), 776–784 (2013)
6. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. *IEEE Security and Privacy* (2008)
7. Pauler, G., Dick, A.: Maximizing profit of a food retailing chain by targeting and promoting valuable customers using Loyalty Card and Scanner Data. *Eur. J. Oper. Res.* **174**(2), 1260–1280 (2006)
8. Pellungrini, R., Pappalardo, L., Pratesi, F., Monreale, A.: A data mining approach to assess privacy risk in human mobility data. Accepted for publication in ACM TIST Special Issue on Urban Computing
9. Pratesi, F., Monreale, A., Trasarti, R., Giannotti, F., Pedreschi, D., Yanagihara, T.: PRISQUIT: a system for assessing privacy risk versus quality in data sharing. Technical report 2016-TR-043. ISTI - CNR, Pisa, Italy (2016)
10. Rizvi, S.J., Haritsa, J.R.: Maintaining data privacy in association rule mining. In: *VLDB 2002* (2002)
11. Rygielski, C., Wang, J.-C., Yen, D.C.: Data mining techniques for customer relationship management. *Technol. Soc.* **24**(4), 483–502 (2002)
12. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information (Abstract). In: *PODS 1998* (1998)
13. Stoneburner, G., Goguen, A., Feringa, A.: Risk management guide for information technology systems: recommendations of the national institute of standards and technology. NIST special publication, vol. 800 (2002)
14. Torra, V.: *Data Privacy: Foundations, New Developments and the Big Data Challenge*. Springer, Heidelberg (2017)

Differential Privacy and Neural Networks: A Preliminary Analysis

Giuseppe Manco and Giuseppe Pirrò^(✉)

ICAR-CNR, Rende, Italy
{manco,pirro}@icar.cnr.it

Abstract. The soaring amount of data coming from a variety of sources including social networks and mobile devices opens up new perspectives while at the same time posing new challenges. On one hand, AI-systems like Neural Networks paved the way toward new applications ranging from self-driving cars to text understanding. On the other hand, the management and analysis of data that fed these applications raises concerns about the privacy of data contributors. One robust (from the mathematical point of view) privacy definition is that of Differential Privacy (DP). The peculiarity of DP-based algorithms is that they do not work on anonymized versions of the data; they add a calibrated amount of noise before releasing the results, instead. The goals of this paper are: to give an overview on recent research results marrying DP and neural networks; to present a blueprint for differentially private neural networks; and, to discuss our findings and point out new research challenges.

1 Introduction

Neural networks (NNs) have recently found many applications in several areas of Artificial Intelligence, ranging from self-driving cars to language understanding. This is both due to significant progresses in terms of techniques [3, 10] to (pre)train complex networks including hundreds of layers (the so-called Deep Networks) involving millions of units, and the availability of large and representative datasets. However, information about individuals maintained by third-parties raises concerns about the protection of user-privacy when analyzing such data. Therefore, it emerges the need for techniques able to offer both utility to the applications and rigorous privacy guarantees.

This problem has been attacked by combining competences from different fields, viz. data mining, cryptography, information hiding, to cite a few. In 2006, Dwork introduced the notion of *differential privacy* [7]. The idea behind differential privacy (DP) can be summarized as follows: *the removal or addition of a single database item does not (substantially) affect the outcome of any analysis*. This can be mathematically guaranteed and thus any data contributor would be more willing to take part into the database as her risk (in terms of privacy violation) does not *substantially* increase. The main challenge is to release aggregate information about the data while protecting the privacy of individual contributors. While DP has been explored in a variety of machine learning and data

mining tasks ranging from regression analysis [21] to classification [2], its usage in neural networks is still in its infancy [1]. The goals of this paper are as follows: (i) providing an overview of techniques that combine DP and Neural Networks; (ii) reporting on our own experience in this field; (iii) discussing pros and cons of our approach and pointing out open problems.

The remainder of the paper is organized as follows. In Sect. 2 we introduce some background. We give an overview on related research in Sect. 3. We describe our approach in Sect. 4. We conclude in Sect. 5.

2 Background

We consider a dataset \mathcal{D} of N tuples $\mathbf{x}_1, \dots, \mathbf{x}_N$ where each tuple $\mathbf{x}_i \in \mathcal{D}$ has $d + q$ attributes $\mathbf{x}_i = \langle x_i^1, \dots, x_i^d, y_i^1, \dots, y_i^q \rangle$. We assume without loss of generality that $\sqrt{\sum_{i=1}^d x_{id}^2} \leq 1$. We model the general data analysis problem by considering a function f , which given a tuple $\mathbf{x} \in \mathcal{D}$ takes as input the tuple's attributes x_1^1, \dots, x_d^d and returns an output y_1^1, \dots, y_q^q as accurate as possible. The function $f(\mathcal{D}, \mathbf{w})$ requires some model parameter \mathbf{w} , and its accuracy is evaluated by using some function $E(\mathcal{D}, \mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{D}} E(\mathbf{x}, \mathbf{w})$ that compares the output of f with the reference values. The goal is to find the optimal parameter models \mathbf{w}^* by solving the optimization problem $\mathbf{w}^* = \underset{\mathbf{w}}{\text{minimize}} E(\mathcal{D}, \mathbf{w})$.

2.1 Neural Networks

In terms of the general learning problem introduced before, the idea is to model the function $f(\mathcal{D}, \mathbf{w})$ via Neural Networks. We will focus on the Multi Layer Perceptron (MLP) [4] model. The MLP is a feed-forward neural network, that is, a network where connections neither are allowed between units in the same layer nor backward. Strictly speaking, a neural network models a nonlinear function from a set of input variables $\mathbf{x}_i, i \in [1, d]$ to a set of output variables $\mathbf{t}_l, l \in [1, q]$ controlled by a vector \mathbf{w} of adjustable parameters.

The topology of a $\langle d, m, q \rangle$ -layer network is given as follows. We assume that each layer $j \in \{1, \dots, m\}$ is characterized by a size s_j , two vectors $\mathbf{z}^{(j)}$ and $\mathbf{a}^{(j)}$ and a matrix $\mathbf{w}^{(j)}$ of weights. We assume that $s_0 = d$ is the size of the input data $\mathbf{x} \in \mathbb{R}^d$ (i.e., number of units in the input layer), and $s_m = q$ is the size of the output (i.e., number of units in the output layer). We refer to ϕ as the activation function for intermediate units and ψ for output units, respectively. The relationships between these components are recursively defined by the following general formulas:

$$\begin{aligned}
z_i^{(0)}(\mathbf{x}, \mathbf{w}) &= x_i \\
z_i^{(j)}(\mathbf{x}, \mathbf{w}) &= \phi \left(a_i^{(j)}(\mathbf{x}, \mathbf{w}) \right) \\
a_i^{(j)}(\mathbf{x}, \mathbf{w}) &= \sum_{k=1}^{n-1} w_{i,k}^{(j)} \cdot z_k^{(j-1)}(\mathbf{x}, \mathbf{w}) \\
y_q(\mathbf{x}, \mathbf{w}) &= \psi \left(a_q^{(m)}(\mathbf{x}, \mathbf{w}) \right)
\end{aligned}$$

To keep notation uncluttered, we leave out bias terms and assume that $S = \sum_{j=1}^m s_j$ and the vectors \mathbf{z} and \mathbf{a} span over \mathbb{R}^S where $a_i \equiv a_v^{(h)}$ for some encoding $i = \langle h, v \rangle$. This induces a partial order $j \prec i$ which holds when $i = \langle h+1, v \rangle$ and $j = \langle h, u \rangle$. Thus, the weight matrices can be represented by a unique vector \mathbf{w} , where $w_{i,j}$ corresponds to the weight of the connection between nodes j and i such that $j \prec i$. This allows us to simplify the previous equations as:

$$\begin{aligned}
z_{\mathbf{x},i} &= \phi_i(a_{\mathbf{x},i}) \\
a_{\mathbf{x},i} &= \sum_{j \prec i} w_{i,j} \cdot z_{\mathbf{x},j}
\end{aligned}$$

where $z_{\mathbf{x},i}$ (resp. $a_{\mathbf{x},i}$) represents the application of a_i to \mathbf{x} , ϕ_i represents the activation function relative to unit i .

Training Neural Networks. After defining the topology, the problem that arises is that of *training the network*, that is, learning the model parameters (weights and biases) that minimize the error function $E(\mathcal{D}, \mathbf{w})$. This is typically done by using some variant of the back-propagation algorithm [13] coupled with pre-training techniques [3, 10] that have been successful for deep networks (i.e., networks including many layers).

2.2 Differential Privacy

Differential privacy captures the increased risk to one's privacy that incurs by participating into a database. It does so by offering to individual participants the guarantees that the output of a query would also have been the same, with sufficiently high probability, even if the record(s) of a participant were not present in the database. Being differentially private is a property of the data access mechanism and is unrelated to the presence or absence of auxiliary information available to the adversary. Differential privacy works by introducing randomness [8].

Proposition 1 ((ϵ, δ)-Differential Privacy). *A randomized function f guarantees (ϵ, δ) -differential privacy if for all data sets \mathcal{D} and \mathcal{D}' differing on at most one element, and all $O \subseteq \text{Range}(f)$, it holds that: $\frac{\Pr[f(\mathcal{D}) \in O]}{\Pr[f(\mathcal{D}') \in O]} \leq e^\epsilon + \delta$*

In the definition, $\epsilon > 0$ is a known privacy parameter that controls the strength of the differential privacy guarantee: larger (resp., smaller) values of ϵ yields weaker (resp., stronger) privacy. δ models the probability that the condition on ϵ fails to hold. The definition of (ϵ, δ) -DP becomes more stringent as ϵ and δ approach 0. Moreover, when ϵ is small, then $e^\epsilon \sim 1 + \epsilon$. Since differential privacy works in an interactive setting, the randomized computation f is the algorithm applied by the data curator when releasing information. When $\delta = 0$ we talk about ϵ -DP; however, (ϵ, δ) -DP behaves better when one needs to *compose* (ϵ, δ) -DP mechanisms e.g., when multiple accesses to the database are required [11].

2.3 Achieving Differential Privacy

We shall now outline how to achieve DP in practice. For real-valued functions, DP can be achieved by adding a calibrated amount of noise to the answer of a query before releasing its results. The amount of noise depends on the *sensitivity* of the function f , that is, how much the answer would change when an arbitrary tuple in the input data is modified.

Definition 2 (Sensitivity). *Given a function $f: \mathcal{D} \rightarrow \mathcal{R}^d$, its sensitivity is:*

$$S(f) = \max_{D \sim D'} \|f(D) - f(D')\|_1.$$

where $\|\cdot\|_1$ is the L_1 norm; $D \sim D'$ means that the database differ in one tuple.

Note that $S(f)$ does not depend on the data but is a property of the function f . $S(f)$ captures the magnitude by which a single individual's data can change the function f in the worst case, and therefore, the *uncertainty in the response* that we must introduce in order to hide the participation of a single individual. The sensitivity of a function gives an upper bound on how much we must perturb its output to preserve privacy. Other notions of sensitivity like *smooth sensitivity* [17] guarantee (ϵ, δ) -DP by relating the amount of noise not only to the query result but also to the database.

The Laplace Mechanism. The Laplace Mechanism is based on the idea of computing the output of a function f , and then perturb each coordinate with noise drawn from the Laplace distribution. The scale of the noise will be calibrated to the sensitivity of f (i.e., $S(f)$) divided by ϵ , which represents the privacy budget.

Proposition 3 (Laplace Mechanism). *For a function $f: \mathcal{D} \rightarrow \mathcal{R}^d$, the mechanism that returns $f(D) + \mathbf{z}$, where each $z_i \in \mathbf{z}$ is drawn from $\text{Lap}(S(f) / \epsilon)$ satisfies ϵ -differential privacy [7].*

The Gaussian Mechanism. The Gaussian mechanism reduces the probability of very large noise while guaranteeing (ϵ, δ) -DP.

Proposition 4 (Gaussian Mechanism). *For a function $f:D \rightarrow \mathcal{R}^d$, the mechanism that returns $f(D) + \mathbf{z}$, where each $z_i \in \mathbf{z}$ is drawn from $\mathcal{N}(0, S(f)^2 \sigma^2)$ ¹ with $\sigma > \exp(-(\sigma\epsilon)^2/2)/1.25$ and $\epsilon < 1$, satisfies (ϵ, δ) -DP [8].*

Non-numeric Queries. For non-numeric queries, achieving DP requires the definition of some utility function to sample one of its outcomes according to the probability distribution close to the optimum. This approach is referred to as the *exponential mechanism* [8].

2.4 Composing Differentially Private Mechanisms

So far we have discussed the general approach to guarantee DP when considering a single access to the data. Nevertheless, one usually needs to perform multiple application of DP-algorithms and thus accessing multiple times to the data with the risk of increasing privacy breaches. Investigating the impact of multiple data accesses requires to understand how DP mechanisms compose. In other words, when accessing databases multiple times via differentially private mechanisms, each of which having its own privacy guarantees, how much privacy is still guaranteed on the union of those outputs? [11].

In what follows we focus on (ϵ, δ) -DP and report on the main research results about composition by considering a scenario involving $T = 10K$ steps (data accesses). To guarantee that each data access is (ϵ, δ) -DP the standard deviation in the Gaussian mechanism should be set as follows: $\sigma = \frac{\sqrt{\log 1/\delta}}{\epsilon}$ [8].

Basic Composition. This is the most simple way of investigating the impact of multiple accesses to the data by the same mechanism. Basically, when one needs to perform T steps, the overall process is $(T\epsilon, T\delta)$ -DP; in other words, the privacy parameters of the individual mechanisms basically sum up [8]. By assuming $\sigma = 4$ and $\delta = 10^{-5}$, each step is $(1.2, 10^{-5})$ -DP and thus after 10K steps the composition gives $(12000, .1)$ -DP.

Advanced Composition. Advanced composition of (ϵ, δ) -DP mechanisms (see e.g., [9]) significantly improve the overall privacy cost. In this case we have that after T steps the process is $(\epsilon\sqrt{T\log 1/\delta}, T\delta)$ -DP. Thus after 10K steps the composition gives $(360, .1)$ -DP.

Amplification by Sampling. The results of advanced composition is further improved by sampling [12]. By considering a batch size q , each step is $(2qe, q\delta)$ -DP. As an example if q is 1% of the data, with $\sigma = 4$ and $\delta = 10^{-5}$ we have that each step is $(.024, 10^{-5})$ -DP. When composing T step we have $(2qe\sqrt{T\log 1/\delta}, qT\delta)$ -DP and thus for $T = 10.000$ the process is $(10, .001)$ -DP.

Optimal Composition. Kairouz et al. [11] recently provide lower bounds for composition. It is previously known that it is sufficient to add Gaussian noise with variance $O(T\Delta^2 \log(1/\delta)/\epsilon^2)$ to each step in order to ensure (ϵ, δ) -DP after T steps. Authors show that this condition can be improved by a log factor. In particular, if each step is (ϵ_0, δ_0) -DP, then the composition of T steps gives

¹ In this case the sensitivity makes usage of the L_2 norm.

$(T\epsilon_0^2 + \sqrt{2T\epsilon_0^2 \log(e + \sqrt{T\epsilon_0^2/\tilde{\delta}})}, T\delta_0 + \tilde{\delta})$ -DP assuming $\epsilon_0 \leq 0.9$ and a slack $\tilde{\delta} > 0$.

Accountant Mechanism. A different approach to keep track of the privacy spent during the course of multiple access to sensitive data makes usage of an *accountant mechanism* [16]. This approach has been recently used by Abadi et al. [1] to train neural networks under DP. Authors leverage the moments of the noise distribution and a bound on the gradient values (via clipping) to obtain a more refined result about the composition than the advanced composition. In this case, the result after T steps and considering a batch size q becomes $(2qe\sqrt{T}, \delta)$ -DP. Thus, we have that for $T = 10.000$ steps and $q = 1\%$ the overall process is $(1.25, 10^{-5})$ -DP.

3 An Overview of the State of the Art

In this section we review applications of DP for Deep Neural Networks by distinguishing between *functional* and *non-functional* approaches.

Functional Approaches. The general idea of differential privacy is to introduce noise. However, it has been observed that it is not easy to adapt this approach in some tasks like logistic regression (e.g., [21]). Chaudhuri et al. [5] came up with a new way of achieving DP in these contexts: *instead of perturbing the results of the regression analysis, one can perturb the objective function and then optimize the perturbed objective function*. The overall idea is reported in Fig. 1.

The original idea of Chaudhuri has been further refined into the Functional Mechanism (FM) [21]. The FM also perturbs the objective function of regression analysis and then release model parameters that minimize such perturbed function. The approach has been applied to both linear and logistic regression; in this latter case by resorting to a second order Taylor expansion of the error function. Authors show that the amount of noise to be injected in order to guarantee DP basically depends on the number of dimensions of the input dataset. The FM has been recently applied to train autoencoders under DP [18].

Non-functional Approaches. Non-functional approaches determine the model parameters that minimize the original loss function instead of its perturbed and approximated version. In the context of neural networks, there are a few attempts at using DP; these approaches are mainly based on *line search methods* such as the (Stochastic) Gradient Descent (SGD). An overview of non-functional approaches is provided in Fig. 1(b). The idea is to start with some initial values of the model parameters and then iteratively update these parameters following the opposite direction of the gradient, after adding noise. We showed in Sect. 2.4 how to determine the amount of noise in the Gaussian mechanism under different types of composition. One of the early approaches in learning via SGD and DP has been proposed by Rajkumar et al. [19]. This approach performs Gaussian perturbation of the overall multiparty objective to achieve (ϵ, δ) -DP. The goal is to compute gradient-based minimization of the objective function in

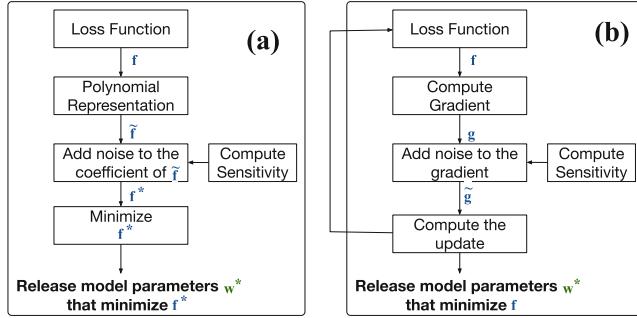


Fig. 1. Functional (a) and non-functional (b) approaches for learning with DP.

a differentially-private way. Privacy is achieved by perturbing the gradient information with *two sources of noise*, the second being needed to avoid an attacker to reconstruct the true minimizer (optimal weights) of the objective function. We already mentioned the work by Abadi et al. [1] that use a SGD-based algorithm and find a tighter bound on the composition of DP mechanisms than previous approaches (e.g., [20]). As a side note, we shall also mention that the functional approach described by Phan et al. [18] to train deep autoencoders also resorts to the SGD to fine tune the parameters.

4 Neural Networks and Differential Privacy

In this section, we describe our experience in applying the functional mechanism to the multilayer perceptron (see Sect. 2.1). Our algorithm involves the following four main phases: (i) find a polynomial representation of the objective function; (ii) compute the sensitivity; (iii) add noise to the polynomial representation; (iv) minimize the perturbed objective function.

Polynomial Representation of the Objective Function. We consider the following 2nd-order Taylor expansion of the error function $E(\mathcal{D}, \mathbf{w}) = \sum_{\mathbf{x} \in \mathcal{D}} E_{\mathbf{x}}(\mathbf{w})$.

$$\hat{E}(\mathcal{D}, \mathbf{w}) \approx E(\mathcal{D}, \hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})^T \nabla E(\mathcal{D}, \hat{\mathbf{w}}) + \frac{1}{2} (\mathbf{w} - \hat{\mathbf{w}})^T \nabla^2 E(\mathcal{D}, \hat{\mathbf{w}}) (\mathbf{w} - \hat{\mathbf{w}})$$

where: $\nabla E(\mathcal{D}, \hat{\mathbf{w}}) \equiv \frac{\partial E}{\partial w_{i,j}}|_{\mathbf{w}=\hat{\mathbf{w}}}$ and $(\nabla^2 E(\mathcal{D}, \hat{\mathbf{w}}))_{i,j} \equiv \frac{\partial^2 E}{\partial w_i \partial w_j}|_{\mathbf{w}=\hat{\mathbf{w}}}$ are the Jacobian and Hessian matrices, respectively. Denoting by $g_{i,j}$ (resp., $h_{i,j}$) an element of the Jacobian (resp., Hessian) matrix, we obtain:

$$g_{i,j} = \sum_{\mathbf{x} \in \mathcal{D}} \delta_{\mathbf{x},i} z_{\mathbf{x},j} \quad (1)$$

$$\delta_{\mathbf{x},i} = z'_{\mathbf{x},i} \sum_{v:i \prec v} \delta_{\mathbf{x},v} w_{v,i} \quad (2)$$

$$h_{i,j} = \sum_{\mathbf{x} \in \mathcal{D}} b_{\mathbf{x},i} z_{\mathbf{x},j}^2 \quad (3)$$

$$b_{\mathbf{x},i} = z''_{\mathbf{x},i} \sum_{v:i \prec v} w_{v,i} \delta_{\mathbf{x},v} + (z'_{\mathbf{x},i})^2 \sum_{v:i \prec v} w_{v,i}^2 b_{\mathbf{x},v} \quad (4)$$

Equations (1) and (3) are defined recursively by considering the values of δ and b of all units. Equations (2) and (4) define such values for internal units. Since the number of model parameters (weights and biases) of the network can be very large computing the whole Hessian matrix is extremely demanding from a computational point of view (it requires a quadratic number operations). Hence, to reduce the computational cost we only consider diagonal elements of the Hessian [6]. As for the output units, their output depends on the specific loss function considered. As an example, when considering the Least Squares Error, we obtain:

$$E(\mathbf{x}, \mathbf{w}) = \frac{1}{2} \sum_{c=1}^d (y_{\mathbf{x},c} - t_{\mathbf{x},c})^2$$

$$y_{\mathbf{x},c} = y_c(\mathbf{x}; \mathbf{w})$$

In the previous formula, $y_{\mathbf{x},c}$ is the c -th component given by the network when giving as input the vector \mathbf{x} and $t_{\mathbf{x},c}$ is the c -th target value. Hence, for output units we have:

$$\delta_{\mathbf{x},i} = (\psi(a_{\mathbf{x},i}) - t_{\mathbf{x},i}) \psi'(a_{\mathbf{x},i}) \quad (5)$$

$$b_{\mathbf{x},i} = (\psi'(a_{\mathbf{x},i}))^2 + (\psi(a_{\mathbf{x},i}) - t_{\mathbf{x},i}) \psi''(a_{\mathbf{x},i}) \quad (6)$$

Compute Sensitivity. In order to determine the amount of noise to be injected into the polynomial representation derived in the previous section we need to estimate its sensitivity. We have the following preliminary result.

Lemma 1. *Let $\mathcal{D}, \mathcal{D}'$ be any two databases differing in at most one tuple, and*

$$\hat{E}(\mathcal{D}, \mathbf{w}) = \left(\sum_{\mathbf{x} \in \mathcal{D}} \sum_{j \prec i} \delta_{\mathbf{x},i} z_{\mathbf{x},j} \right) w_{i,j} + \frac{1}{2} \left(\sum_{\mathbf{x} \in \mathcal{D}} \sum_{j \prec i} b_{\mathbf{x},i} z_{\mathbf{x},j}^2 \right) w_{i,j}^2$$

$$\hat{E}(\mathcal{D}', \mathbf{w}) = \left(\sum_{\mathbf{x}' \in \mathcal{D}'} \sum_{j \prec i} \delta_{\mathbf{x}',i} z_{\mathbf{x}',j} \right) w_{i,j} + \frac{1}{2} \left(\sum_{\mathbf{x}' \in \mathcal{D}'} \sum_{j \prec i} b_{\mathbf{x}',i} z_{\mathbf{x}',j}^2 \right) w_{i,j}^2$$

the polynomial representation of the error function on \mathcal{D} and \mathcal{D}' , respectively. Let \mathbf{x} be an arbitrary tuple. We have that the sensitivity is:

$$\|\hat{E}(\mathcal{D}, \mathbf{w}) - \hat{E}(\mathcal{D}', \mathbf{w})\|_1 \leq 2 \sum_{j \prec i} \max_{\mathbf{x}} (|\delta_{\mathbf{x},i} z_{\mathbf{x},j}| + |b_{\mathbf{x},i} z_{\mathbf{x},j}^2|)$$

Adding Noise. After determining the sensitivity, and thus the amount of noise, we shall now provide an algorithm for learning with DP. The algorithm is sketched in Algorithm 1. It starts with an initialization of the weights (line 1) and then requires to obtain the Jacobian and (approximated) Hessian matrices representing the building blocks of the polynomial representation of the error function (line 2). At line 3 the noise is injected into such coefficients to obtain a perturbed error function, which is then minimized (according to one of the existing methods) in line 4. The final step consists in releasing the model parameters (the set of weights $\tilde{\mathbf{w}}$) that minimize it.

Theorem 5. *Algorithm 1 satisfies ϵ -differential privacy.*

Proof. Consider two neighbor datasets \mathcal{D} and \mathcal{D}' that differ on the last tuple, \mathbf{x}_n and \mathbf{x}'_n for \mathcal{D} and \mathcal{D}' , respectively. The proof proceeds by applying the definition of differential privacy (see Definition 1).

$$\begin{aligned} \frac{\Pr\{\tilde{E}(\mathbf{w}) \mid \mathcal{D}\}}{\Pr\{\tilde{E}(\mathbf{w}) \mid \mathcal{D}'\}} &= \frac{\exp\left(\frac{\epsilon \|\sum_{\mathbf{x} \in \mathcal{D}} \sum_{j \prec i} (\delta_{\mathbf{x},i} z_{\mathbf{x},j} + b_{\mathbf{x},i} z_{\mathbf{x},j}^2) - (\delta_i z_j + b_i z_j^2)\|_1}{S(\tilde{E})}\right)}{\exp\left(\frac{\epsilon \|\sum_{\mathbf{x}' \in \mathcal{D}'} \sum_{j \prec i} (\delta_{\mathbf{x}',i} z_{\mathbf{x}',j} + b_{\mathbf{x}',i} z_{\mathbf{x}',j}^2) - (\delta_i z_j + b_i z_j^2)\|_1}{S(\tilde{E})}\right)} \\ &\leq \exp\left(\frac{\epsilon}{S(\tilde{E})} \|\sum_{\mathbf{x} \in \mathcal{D}} \sum_{j \prec i} (\delta_{\mathbf{x},i} z_{\mathbf{x},j} + b_{\mathbf{x},i} z_{\mathbf{x},j}^2) - \sum_{\mathbf{x}' \in \mathcal{D}'} \sum_{j \prec i} (\delta_{\mathbf{x}',i} z_{\mathbf{x}',j} + b_{\mathbf{x}',i} z_{\mathbf{x}',j}^2)\|_1\right) \\ &= \exp\left(\frac{\epsilon}{S(\tilde{E})} \|\sum_{j \prec i} (\delta_{\mathbf{x}_n,i} z_{\mathbf{x}_n,j} + b_{\mathbf{x}_n,i} z_{\mathbf{x}_n,j}^2) - \sum_{j \prec i} (\delta_{\mathbf{x}'_n,i} z_{\mathbf{x}'_n,j} + b_{\mathbf{x}'_n,i} z_{\mathbf{x}'_n,j}^2)\|_1\right) \\ &\leq \exp\left(\frac{\epsilon}{S(\tilde{E})} 2 \sum_{j \prec i} \max_{\mathbf{x}} (|\delta_{\mathbf{x},i} z_{\mathbf{x},j}| + |b_{\mathbf{x},i} z_{\mathbf{x},j}^2|)\right) \text{(by Lemma 1)} \\ &\leq \exp(\epsilon) \end{aligned}$$

Algorithm 1. FunctionalNetDP (Privacy budget ϵ)

- 1: Initialize \mathbf{w}
 - 2: $g, h \leftarrow$ Polynomial representation of the loss function
 - 3: Find \tilde{g} and \tilde{h} via **addNoise**(\mathbf{w}, ϵ) /* Algorithm 2 */
 - 4: Compute $\tilde{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \tilde{E}(\mathcal{D}, \mathbf{w})$
 - 5: **return** $\tilde{\mathbf{w}}$ / set of weights that minimizes $\tilde{E}(\mathcal{D}, \mathbf{w})$
-

Algorithm 2. addNoise(Weights \mathbf{w} , privacy budget ϵ)

- 1: Let \mathcal{D} be the dataset
 - 2: Let $S(\tilde{E})$ be the sensitivity of the polynomial representation fo the network.
 - 3: Find g via Eq. (1) and h via Eq. (3) using \mathcal{D}
 - 4: $g \leftarrow g/(1, \|g\|^2/C)$ /* clip Gradient */
 - 5: $h \leftarrow h/(1, \|h\|^2/C)$ /* clip Hessian */
 - 6: $\tilde{g} \leftarrow g + \text{Lap}(S(\tilde{E}) \mid \epsilon)$
 - 7: $\tilde{h} \leftarrow h + \text{Lap}(S(\tilde{E}) \mid \epsilon)$
 - 8: **return** \tilde{g} and \tilde{h}
-

4.1 Estimating the Amount of Laplacian Noise

Algorithm 1 lays the foundation for achieving differential privacy. We have shown in Lemma 1 that the amount of noise depends on the maximum (over all tuples) sum of the coefficients of δ , z , and b . In turn, these terms depend on the choice of the activation functions for the intermediate units (i.e., ϕ), output units (i.e., ψ) and the value of weights (i.e., $w_{i,j}$). We now provide a finer grained estimation of the noise. Assuming that for both ϕ and ψ the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$ is used, then we have that $\sigma(x) \in [0, 1]$, $\sigma'(x) \in [0, 0.25]$, and $\sigma(x)'' \in [-0.1, 0.1]$.

Moreover, we assume that each $w_{i,j} \in [0, 1]$ and that for each dimension d of the input tuple $\mathbf{x}^d \in [0, 1]$. At this point, to bound the amount of Laplace noise needed, we need to bound the components $\sum_{j < i} |\delta_{\mathbf{x},i} z_{\mathbf{x},j}|$ and $\sum_{j < i} |b_{\mathbf{x},i} z_{\mathbf{x},j}^2|$ in Lemma 1. Let m be the depth of the network. For each layer $j \in \{1, \dots, m\}$ we denote by δ^j (resp., b^j) the coefficient of a generic unit in the layer j . Moreover, s_j represents the number of units at layer j .

Lemma 2. *The noise that the $\sum_{j < i} |\delta_{\mathbf{x},i} z_{\mathbf{x},j}|$ component in Lemma 1 contributes is $\leq 0.25^m \times \prod_{j=1}^{j=m} s_j$.*

Proof. The idea is to analyze the layer-wise upperbound of δ and z when considering a generic tuple \mathbf{x} . As for z we always have $z \in [0, 1]$ (since we are using the sigmoid activation function). As for δ we have:

$$\delta \leq \begin{cases} 0.25 & \text{if } j = m \quad (\text{Eq. (5)}) \\ 0.25^{m-j+1} \times \prod_{q=j}^{q=m} s_q & j \neq m \quad (\text{Eq. (2)}) \end{cases}$$

The maximum contribution occurs when $j = 1$ in $\prod_{q=j}^{q=m} s_q$.

Lemma 3. *The contribution of the $\sum_{j < i} |b_{\mathbf{x},i} z_{\mathbf{x},j}^2|$ component in Lemma 1 is:*

$$\leq \prod_{j=1}^{j=m} s_j (0.1^m + 0.001625 \times m)$$

Proof. Similarly to the previous case, we have that:

$$b \leq \begin{cases} 0.1625 & \text{if } j = m \quad (\text{Eq. (6)}) \\ 0.1^{m-j+1} \times \prod_{q=j}^{q=m} s_q \\ + (0.1)^2 (m-j) 0.1625 \prod_{q=j}^{q=m} s_q & j \neq m \quad (\text{Eq. (4)}) \end{cases}$$

The contribution is maximum when $j = 1$.

Theorem 6. *For a network of m layers with s_j units in each layer, where $j \in [1, n]$, the amount of Laplacian noise to ensure differential privacy is:*

$$\leq 2 \left(0.25^m \prod_{q=1}^{q=m} s_q + \prod_{j=1}^{j=m} s_j \times (0.1^m + 0.001625 \times m) \right)$$

Proof. The idea is to substitute the bounds in the previous lemmas in Lemma 1.

The above analysis shows that the amount of noise to ensure differential privacy is dominated by the number of units in the network.

5 Concluding Remarks and Future Work

We have reported on the usage of differential privacy in neural networks and discussed our preliminary findings in adding Laplacian noise to a low-polynomial representation of the error function. We now discuss some crucial aspects of our approach. *First*, we have shown via Lemma 1 that the sensitivity basically depends on the topology of the network. The result about the sensitivity when using the functional mechanism for neural networks is in stark contrast with functional approaches tackling regression (e.g., [21]); in that case the sensitivity was bound by the dimensionality of the input data. This comes as no surprise because of the fact that our approach involves a complex topology of interconnected network units (neurons) while regression only makes usage of a single unit, in a sense. To mitigate the impact of sensitivity we bound in Algorithm 2 the values of the gradient and Hessian (lines 5 and 6) before adding noise. Finding a tighter bound for Lemma 1 is in our research agenda.

Second, we pursued the simplest way to ensure DP by adding Laplacian noise to the gradient and the Hessian coefficients (the coefficients of our polynomial representation), and use the noisy version to perform the minimization (line 5 Algorithm 1). We note that the sensitivity in our case is irrelevant to the size of the data set at hand. At the same time we note that the potential effect of the noise can have different effects on the gradient and Hessian, respectively. In this latter case, even a little amount of noise in the coefficients of the Hessian matrix can lead to large changes on the parameters being updated, with the extreme potential consequence that the Hessian matrix's positive definiteness is destroyed, which implies that a global optimal solution may not be attained at all. One way to approach the problem would be to threshold the eigenvalues of the Hessian to ensure positive definiteness; nevertheless, this method might generate coefficients that are far from the original (pre-noise) coefficients. Another approach could be to leverage public data sets for the Hessian, that is, datasets where participants voluntarily disclosed information and we use public and private data sets to compute the gradient.

Third, Algorithm 1 works by approximating the Hessian to its diagonal components as done by other approaches (e.g., [14]). Nevertheless, in some cases, the approximation can be very rough leading to poor solutions. This problem may be exacerbated by the noise addition. One way to overcome this problem could be to resort to non-functional approaches.

In this context, an interesting aspect that has not yet been investigated in the context of DP is the usage of 2nd order methods like the Hessian Free (HF) technique [15]. The idea of HF is to approximate the error function around each parameter up to second order (SGD-based methods only take into account the

first derivatives of the parameters). Operationally, the HF approach requires gradient estimations, approximations of the Hessian (e.g., via Gauss-Newton approximation) and its minimization. This latter step is usually done by using the conjugate gradient (CG). The CG applies minimization along each dimension of the parameter space separately and thus would require a number of iterations equals to the number of dimensions of the parameter space to converge in general. However, it usually makes significant progress toward a minimum in a much lower number of iterations [15]. We see a number of research challenges for the usage of DP in this setting, among which: (i) investigating the amount of noise needed along each iteration of the HF and the properties of composition; and (ii) comparing this approach with first-order methods (e.g., [1]).

Fourth, in this preliminary analysis, we have discussed privacy without considering utility. Our ongoing experimental evaluation shows that while our approach achieves strong privacy guarantees, when the size of the network grows² the utility is heavily affected. To mitigate this issue, we are evaluating the impact of privacy on utility when considering the HF technique.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of CCS, pp. 308–318. ACM (2016)
2. Aggarwal, C.C., Philip, S.Y. (eds.): Privacy-Preserving Data Mining. Springer, Heidelberg (2008)
3. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems 19, p. 153 (2007)
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
5. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: Advances in Neural Information Processing Systems, pp. 289–296 (2009)
6. Cun, Y.L.: Modeles Connexionistes de l’Apprentissage. Ph.D. thesis, Université de Paris (1987)
7. 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). https://doi.org/10.1007/11787006_1
8. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3–4), 211–407 (2014)
9. Dwork, C., Rothblum, G.N., Vadhan, S.: Boosting and differential privacy. In: Proceedings of FOCS, pp. 51–60. IEEE (2010)
10. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
11. Kairouz, P., Oh, S., Viswanath, P.: The composition theorem for differential privacy. In: ICML, pp. 1376–1385 (2015)
12. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? SIAM J. Comput. **40**(3), 793–826 (2011)

² Anecdotally, when considering more than 4 layers with hundreds of units per layer.

13. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. *JMLR* **10**, 1–40 (2009)
14. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient BackProp. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade*. LNCS, vol. 7700, pp. 9–48. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_3
15. Martens, J.: Deep learning via Hessian-free optimization. In: *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pp. 735–742 (2010)
16. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: *Proceedings of SIGMOD*, pp. 19–30. ACM (2009)
17. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: *Proceedings of STOC*, pp. 75–84. ACM (2007)
18. Phan, N., Wang, Y., Wu, X., Dou, D.: Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI*, pp. 12–17 (2016)
19. Rajkumar, A., Agarwal, S.: A differentially private stochastic gradient descent algorithm for multiparty classification. In: *International Conference on Artificial Intelligence and Statistics*, pp. 933–941 (2012)
20. Song, S., Chaudhuri, K., Sarwate, A.D.: Stochastic gradient descent with differentially private updates. In: *2013 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 245–248. IEEE (2013)
21. Zhang, J., Zhang, Z., Xiao, X., Yang, Y., Winslett, M.: Functional mechanism: regression analysis under differential privacy. *Proc. VLDB Endow.* **5**(11), 1364–1375 (2012)

Co-clustering for Differentially Private Synthetic Data Generation

Tarek Benkhelif^{1,2(✉)}, Françoise Fessant¹, Fabrice Clérot¹,
and Guillaume Raschia²

¹ Orange Labs, 2, Avenue Pierre Marzin, 22307 Lannion Cédex, France
{tarek.benkhelif,francoise.fessant,fabrice.clerot}@orange.com

² LS2N - Polytech Nantes,
Rue Christian Pauc, BP 50609, 44306 Nantes Cédex 3, France
guillaume.raschia@univ-nantes.fr

Abstract. We propose a methodology to anonymize microdata (i.e. a table of n individuals described by d attributes). The goal is to be able to release an anonymized data table built from the original data while meeting the differential privacy requirements. The proposed solution combines co-clustering with synthetic data generation to produce anonymized data. First, a data independent partitioning on the domains is used to generate a perturbed multidimensional histogram; a multidimensional co-clustering is then performed on the noisy histogram resulting in a partitioning scheme. This differentially private co-clustering phase aims to form attribute values clusters and thus, limits the impact of the noise addition in the second phase. Finally, the obtained scheme is used to partition the original data in a differentially private fashion. Synthetic individuals can then be drawn from the partitions. We show through experiments that our solution outperforms existing approaches and we demonstrate that the produced synthetic data preserve sufficient information and can be used for several datamining tasks.

Keywords: Differential privacy · Co-clustering
Synthetic individual data

1 Introduction

There is an increasingly social and economic demand for open data in order to improve planning, scientific research or market analysis. In particular, the public sector via its national statistical institutes, healthcare or transport authorities, is pushed to release as much information as possible for the sake of transparency. Private companies are also implicated in the valorization of their data through exchange or publication. Orange has recently made available to the scientific community several mobile communication datasets collected from its networks in Senegal and Ivory Coast as part of D4D challenges (Data for Development). These challenges have shown the potential added-value of analyzing such data

for several application domains which address both development projects and improvement of public policies effectiveness [3]. This demand for publicly available data motivated the research community to propose several privacy preserving data publishing solutions.

Problem Statement. The literature about privacy preserving data publishing is mainly organized around two privacy concepts (i) group anonymization techniques such as k -anonymity [13] and (ii) random perturbation methods with in particular the concept of Differential Privacy (DP) [6]. K -anonymity seeks to prevent re-identification of records by making each record indistinguishable within a group of k or more records and allows the release of data in its original form. The notion of protection defended by DP is the strong guarantee that the presence or absence of an individual in a dataset will not significantly affect the result of aggregated statistics computed from this dataset. DP works by adding some controlled noise to the computed function. There are two models for differential privacy: the interactive model and the non-interactive model. A trusted third party collects data from data owners and make it available for data users. In the interactive model, the trusted party catches the queries sent by data users and outputs a sanitized response. In the non-interactive model, the trusted party publishes a protected version of the data. In this paper, we study the problem of differentially private data generation. We consider the non-interactive model and seek to release synthetic data, providing utility to the users while protecting the individuals represented in the data.

Contributions. We present an original differentially private approach that combines co-clustering, an unsupervised data mining analysis technique, and synthetic data generation. We summarize our contributions below.

- We study and implement a two-phase co-clustering based partitioning strategy for synthetic data generation.
- We experimentally evaluate the released data utility, by measuring the statistical properties preservation and the predictive performance of the synthetic data.
- We compare our approach with other existing differentially private data release algorithms.

The paper is organized as follows. Section 2 first identifies the most related efforts to our work, Sects. 3 and 4 give the necessary background on differential privacy and co-clustering, in Sect. 5 the proposed approach is described. The utility of the produced synthetic datasets is evaluated in Sect. 6. The final section gathers some conclusions and future lines of research.

2 Related Work

There are many methods designed for learning specific models with differential privacy, but we briefly review here the most related approaches to our work, and we only focus on histogram and synthetic data generation.

The first propositions that started addressing the non-interactive data release while achieving differential privacy are based on histogram release. Dwork et al. [7] proposed a method that publishes differentially private histograms by adding a Laplacian random noise to each cell count of the original histogram, it is considered as a baseline strategy. Xu et al. [15] propose two approaches for the publication of differentially private histograms: NoiseFirst and StructureFirst. NoiseFirst is based on the baseline strategy: a Laplacian random noise is first added to each count as in [7]. It is followed by a post-optimization step in which the authors use a dynamic programming technique to build a new histogram by merging the noisy counts. StructureFirst consists in constructing an optimal histogram using the dynamic programming technique to determine the limits of the bins to be merged. The structure of this optimal histogram is then perturbed via an exponential mechanism. And finally the averages of the aggregated bins are perturbed using the Laplacian mechanism. The authors in [2] propose a method that uses a divisible hierarchical clustering scheme to compress histograms. The histogram bins belonging to the same cluster have similar counts, and hence can be approximated by their mean value. Finally, only the noisy cluster centers, which have a smaller sensitivity are released. All the mentioned contributions deal only with unidimensional and bidimensional histogram publication and are not adapted to the release of multidimensional data. The closest approach to our work is proposed in [14], first, a cell-based partitioning based on the domains is used to generate a fine-grained equi-width cell histogram. Then a synthetic dataset D_c is released based on the cell histogram. Second, a multidimensional partitioning based on kd-tree is performed on D_c to obtain uniform or close to uniform partitions. The resulted partitioning keys are used to partition the original database and obtain a noisy count for each of the partitions. Finally, given a user-issued query, an estimation component uses either the optimal histogram or both histograms to compute an answer of the query. Other differentially private data release solutions are based on synthetic data generation [10, 16]. The proposed solution in [10] first probabilistically generalizes the raw data and then adds noise to guarantee differential privacy. Given a dataset D , the approach proposed in [16] constructs a Bayesian network N , that approximates the distribution of D using a set P of low dimensional marginals of D . After that, noise is injected into each marginal in P to ensure differential privacy, and then the noisy marginals and the Bayesian network are used to construct an approximation of the data distribution in D . Finally, tuples are sampled from the approximate distribution to construct a synthetic dataset that is released. Our work focuses on releasing synthetic data and complements the efforts of [14, 16] in the way that we also study a differentially private aggregation of multidimensional marginals. As in [14], we use the co-clustering like a multidimensional partitioning that is data-aware. And, unlike the variance threshold used in [14] or the θ parameter that determines the degree of the Bayesian network in [16] our solution is parameter-free.

3 Preliminaries and Definitions

3.1 Differential Privacy

Definition 1 (ε -Differential Privacy [5]). A random algorithm \mathcal{A} satisfies ε -differential privacy, if for any two datasets D_1 and D_2 that differ only in one tuple, and for any outcome O of \mathcal{A} , we have

$$\Pr[\mathcal{A}(D_1) = O] \leq e^\varepsilon \times \Pr[\mathcal{A}(D_2) = O], \quad (1)$$

where $\Pr[\cdot]$ denotes the probability of an event.

Laplace Mechanism. To achieve differential privacy, we use the Laplace mechanism that adds random noise to the response to a query. First, the true value of $f(D)$ is computed, where f is the query function and D the data set, then a *random* noise is added to $f(D)$. And the $\mathcal{A}(D) = f(D) + \text{noise}$ response is finally returned. The amplitude of the noise is chosen as a function of the biggest change that can cause one tuple on the output of the query function. This amount defined by Dwork is called sensitivity.

Definition 2 (L_1 -sensitivity). The L_1 -sensitivity of $f : D \rightarrow \mathbb{R}^d$ is defined as

$$\Delta(f) = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_1 \quad (2)$$

For any two datasets D_1 and D_2 that differ only in one tuple.

The density function of the Laplace distribution is defined as follows.

$$\text{Lap}(x|\mu, b) = \frac{1}{2b} \exp\left(\frac{-|x - \mu|}{b}\right) \quad (3)$$

Where μ is called the position parameter and $b > 0$ the scale parameter.

The use of a noise drawn from a Laplacian distribution, $\text{noise} = \text{Lap}(\Delta f/\varepsilon)$, with the position parameter = 0, and the scale parameter = $\Delta f/\varepsilon$ guarantees the ε -differential privacy [11].

Composition. For a sequence of differentially private mechanisms, the composition of the mechanisms guarantees privacy in the following way.

Definition 3 (Sequential composition [9]). For a sequence of n mechanisms $\mathcal{A}_1, \dots, \mathcal{A}_n$ where each \mathcal{A}_i respects the ε_i -differential privacy, the sequence of the \mathcal{A}_i mechanisms ensures the $(\sum_{i=1}^n \varepsilon_i)$ -differential privacy.

Definition 4 (Parallel composition [9]). If D_i are disjoint sets of the original database and \mathcal{A}_i is a mechanism that ensures the ε -differential privacy for each D_i , then the sequence of \mathcal{A}_i ensures the ε -differential privacy.

3.2 Data Model

We focus on microdata. Each record or row is a vector that represents an entity and the columns represent the entity's attributes. We suppose that all the d attributes are nominal or discretized. We use d -dimensional histogram or data cube, to represent the aggregate information of the data set. The records are the points in the d -dimensional data space. Each cell of a data cube represents the count of the data points corresponding to the multidimensional coordinates of the cell.

3.3 Utility Metrics

Hellinger Distance. In order to measure the utility of the produced data, we use the Hellinger distance between the distributions in the original data and our synthetic data. We considered the Kullback-Leibler divergence, but we found the Hellinger distance to be more robust given that multidimensional histograms are highly sparse.

Definition 5 (Hellinger distance). The Hellinger distance between two discrete probability distributions $P = (p_1, \dots, p_k)$ and $Q = (q_1, \dots, q_k)$ is given by:

$$D_{\text{Hellinger}}(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^k (\sqrt{p_i} - \sqrt{q_i})^2}.$$

Random Range Queries. We use random range queries as a utility measure of the synthetic data. We generate random count queries with random query predicates over all the attributes:

Select COUNT(*) From D Where $X_1 \in I_1$ and $X_2 \in I_2$ and ... and $X_d \in I_d$.

For each attribute X_i , I_i is a random interval generated from the domain of X_i . We use the relative error to measure the accuracy of a query q , where $A_{\text{original}}(q)$ denotes the true answer of q on the original data and $A_{\text{perturbed}}(q)$ is the noisy count computed when the synthetic data generated from a differentially private mechanism are used.

Definition 6 (Relative error). $\text{RelativeError}(q) = \frac{|A_{\text{perturbed}}(q) - A_{\text{original}}(q)|}{A_{\text{original}}(q)}$

4 Co-clustering

Co-clustering is an unsupervised data mining analysis technique which aims to extract the existing underlying block structure in a data matrix [8]. The data studied in the co-clustering problems are of the same nature as the data processed by the clustering approaches: they are composed of m observations without label, described by several variables, denoted $\{X_1, X_2, \dots, X_d\}$. These variables can be continuous or nominal, then taking a finite number of different values. The values taken by the descriptive variables are partitioned in order

to obtain new variables $\{X_1^M, X_2^M, \dots, X_d^M\}$ that are called variables-partitions. The values of these new variables are the clusters obtained by the partitions of the values of the variables $\{X_1, X_2, \dots, X_d\}$. Each of the X_i^M variables has $\{k_1, k_2, \dots, k_d\}$ values which are groups of values if the variable is nominal and intervals if the variable is continuous. The MODL approach makes it possible to achieve a co-clustering on the values of d descriptive variables of the data, we will use this particular feature in our work.

4.1 MODL Co-clustering

We choose the MODL co-clustering [4] because: First, MODL is theoretically grounded and exploits an objective Bayesian approach [12] which turns the discretization problem into a task of model selection. The Bayes formula is applied by using a hierarchical and uniform prior distribution and leads to an analytical criterion which represents the probability of a model given the data. Then, this criterion is optimized in order to find the most probable model given the data. The number of intervals and their bounds are automatically chosen. Second, MODL is a nonparametric approach according to Robert [12]: the number of modeling parameters increases continuously with the number of training examples. Any joint distribution can be estimated, provided that enough examples are available.

Data Grid Models. The MODL co-clustering approach allows one to automatically estimate the joint density of several (numerical or categorial) variables, by using a data grid model [4]. A data grid model consists in partitioning each numerical variable into intervals, and each categorical variable into groups. The cross-product of the univariate partitions constitutes a data grid model, which can be interpreted as a nonparametric piecewise constant estimator of the joint density. A Bayesian approach selects the most probable model given the dataset, within a family of data grid models. In order to find the best M^* model (knowing the data D), the MODL co-clustering uses a Bayesian approach called Maximum A Posteriori (MAP). It explores the space of models by minimizing a Bayesian criterion, called *cost*, which makes a compromise between the robustness of the model and its precision:

$$\text{cost}(M) = -\log(P(M|D))\alpha - \log(P(M) * P(D|M)) \quad (4)$$

The MODL co-clustering also builds a hierarchy of the parts of each dimension using an ascending agglomerative strategy, starting from M^* , the optimal grid result of the optimization procedure up to M_\emptyset , the Null model, the unicellular grid where no dimension is partitioned. The hierarchies are constructed by merging the parts that minimize the dissimilarity index $\Delta(c_1, c_2) = \text{cost}(M_{c_1 \cup c_2}) - \text{cost}(M)$, where c_1, c_2 are two parts of a partition of a dimension of the grid M and $M_{c_1 \cup c_2}$ the grid after fusion of c_1 and c_2 . In this way the fusion of the parts minimizes the degradation of the cost criterion, and thus, minimizes the loss of information.

5 DPCocGen

We present our DPCocGen algorithm, a two-phase co-clustering based partitioning strategy for synthetic data generation. First, a data independent partitioning on the domains is used to generate a multidimensional histogram, the Laplace mechanism is used as in the baseline strategy [7] to perturb the histogram. Then, a multidimensional MODL co-clustering is performed on the noisy histogram. This first phase corresponds to a differentially private co-clustering (as shown in Fig. 1) and aims to produce a partitioning scheme. In the second phase, DPCocGen uses the partitioning scheme to partition the original data and computes a noisy count for each of the partitions (using Laplace mechanism). Finally, the noisy counts are used to draw synthetic individuals from each partition.

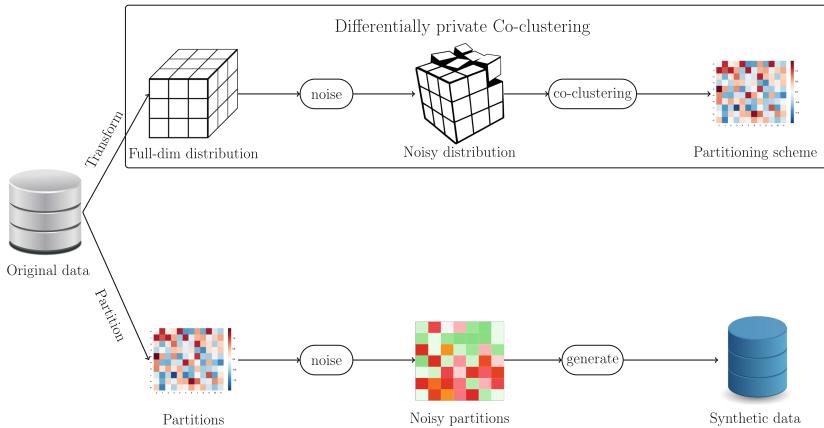


Fig. 1. DPCocGen: a two-phase co-clustering based partitioning strategy for synthetic data generation.

The advantage of this approach lies in the fact that the partitioning scheme obtained through the co-clustering is indirectly dependent on the data structure, the intuition is that even after perturbing the multidimensional histogram, the co-clustering phase will preserve some of the relations between the clusters of attribute values (partitions) of the various dimensions. The resulting cell fusions limit the impact of the noise addition in the second phase. The original data is not consulted during the co-clustering construction which saves the privacy budget that is divided between the two phases to perturb the counts. The detailed steps of DPCocGen are given in Algorithm 1.

5.1 Privacy Guarantee

DPCocGen follows the compositability property of the differential privacy, the first and second phases require direct access to the database, Steps 3 and 7 of the

Algorithm 1. DPCocGen algorithm

Require: Dataset D , the overall privacy budget ε

- 1: **Phase 1:**
 - 2: Build a multidimensional histogram from D .
 - 3: **Perturb** the counts of each cell using a privacy budget ε_1 .
 - 4: Perform a multidimensional co-clustering from the histogram obtained in step 3.
 - 5: **Phase 2:**
 - 6: Partition the data set D based on the partitioning scheme obtained from step 4.
 - 7: **Perturb** the aggregated counts of each partition returned form step 6 using a privacy budget $\varepsilon_2 = \varepsilon - \varepsilon_1$.
 - 8: Generate synthetic individuals from each partition using the perturbed counts returned from step 7 to build a synthetic dataset D' .
-

Algorithm 2. Perturb algorithm

Require: Count c , privacy budget ε

- 1: $c' = c + Lap(1/\varepsilon)$
 - 2: **if** $c' < 0$ **then**
 - 3: $c' = 0$
 - 4: **end if**
 - 5: Return c'
-

Algorithm 1 are $\varepsilon_1, \varepsilon_2$ -differentially private. No access to the original database is invoked during the sampling phase. The sequence is therefore ε -differentially private with $\varepsilon = \varepsilon_1 + \varepsilon_2$.

6 Experiments

In this section we conduct three experiments on a real-life microdata set in order to illustrate the efficiency of our proposition on a practical case. The objective is to explore the utility of synthetic data by measuring the statistical properties preservation, the relative error on a set of random range queries answers and their predictive performance.

6.1 Experimental Settings

Dataset. We experiment with the Adult database available from the UCI Machine Learning Repository¹ which contains 48882 records from the 1994 US census data. We retain the attributes {age, workclass, education, relationship, sex}. We discretize continuous attributes into data-independent equi-width partitions.

¹ <https://archive.ics.uci.edu/ml/>.

Baseline. We implement the baseline strategy [7] to generate a synthetic dataset, a multidimensional histogram is computed and then disturbed through a differentially private mechanism. Records are then drawn from the noisy counts to form a data set.

PrivBayes. We use an implementation of PriveBayes [16] available at [1] in order to generate a synthetic dataset, we use $\theta = 4$ as suggested by the authors.

Privacy Budget Allocation. The privacy budget is equally divided between the two phases of DPCocGen for all the experiments, $\varepsilon_1 = \varepsilon_2 = \varepsilon/2$.

6.2 Descriptive Performance

In this experiment, we are interested in preservation of the joint distribution of the original dataset in the generated synthetic data. In order to measure the difference between two probability distribution vectors we choose the Hellinger distance. First, we compute the multivariate distribution vector P of the original dataset, then, we compute the multivariate distribution vector Q of the synthetic data generated using *DPCocGen* and the multivariate distribution vector Q' of the synthetic data generated using *Base line*. Finally, the distances $D_{\text{Hellinger}}(P, Q)$ and $D_{\text{Hellinger}}(P, Q')$ are measured. For each configuration the distances are calculated through 50 synthetic data sets and represented in Fig. 2. We use box-plots diagrams to represent these results where the x-axis represents the synthetic data generation method, the first box in the left represents the baseline strategy, the following boxes correspond to *DPCocGen* with different levels of granularity (number of cells). The y-axis indicates the Hellinger distance measured between the distribution calculated on the generated data and the original distribution.

Regardless of the privacy budget, the joint probability distribution of the synthetic data generated with *DPCocGen* is closer to the original distribution than the distribution of the data that is obtained using *Baseline*, except when $\varepsilon = 0.5$ and for the *DPCocGen* case with a high co-clustering aggregation level (144 cells), in that particular configuration the partitioning was too coarse and failed to correctly describe the data. The optimal aggregation level varies according to noise magnitude, but the finest aggregation level seems to offer a satisfying result for each configuration.

6.3 Random Range Queries

The goal of this experiment is to evaluate the utility of the produced data in terms of relative error when answering random range queries. We first generate 100 random queries. We produce synthetic datasets using *Base line*, *PrivBayes* and *DPCocGen*. We compute all the queries and report their average error over 15 runs. We use for this experiment the finer co-clustering level. Figure 3 shows that the average relative error decreases as the privacy budget ε grows for the

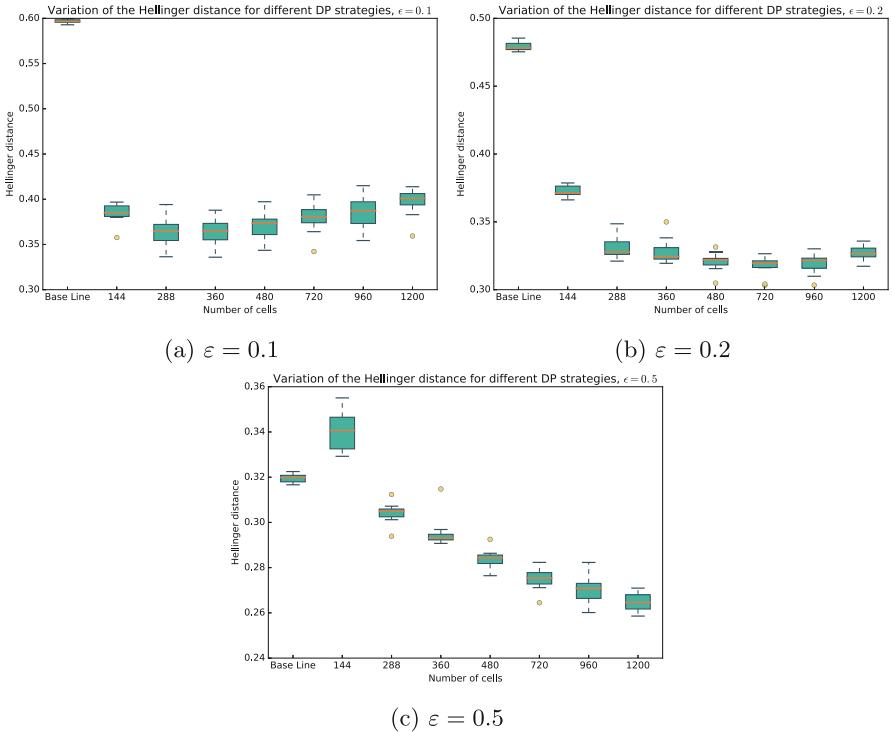


Fig. 2. Joint distribution distances

three algorithms. One can also observe that *PrivBayes* and *DPCocGen* are close and do better than *Base line* regardless of the privacy budget.

6.4 Predictive Performance

In this experiment we are interested in the classification performances obtained with a supervised classifier whose learning is based on synthetic datasets. We randomly select 80% of the observations in order to generate the synthetic data using *DPCocGen*, *Base line* and *PrivBayes*, we use the generated data to train a classifier in order to predict the value of the attributes *Sex* and *Relationship*. The remaining 20% are used for the evaluations. We use for this experiment the finer co-clustering level. The results are presented Figs. 4 and 5, they represent the average on 50 runs. The privacy budget value is shown on the x-axis, the y-axis shows the area under the ROC curve (AUC) measured on the test set. The figure also indicates the performances obtained when the real data are used for learning the model (Original Data).

We retain that the classification performances obtained with *DPCocGen* are close to those obtained when the real data are used for learning the model. The performances of *DPCocGen* are always higher than those of the *Base line* and *PrivBayes*.

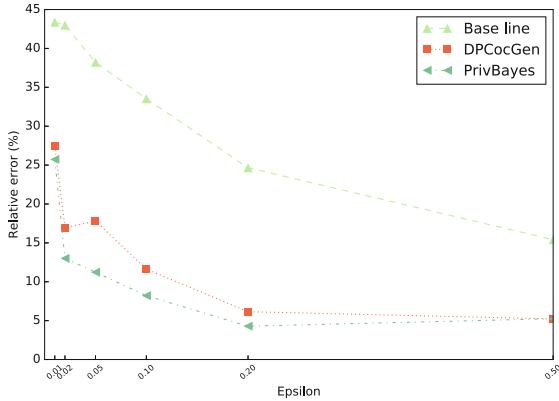


Fig. 3. Random range queries

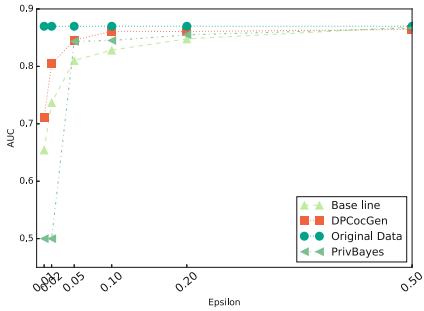


Fig. 4. Sex prediction

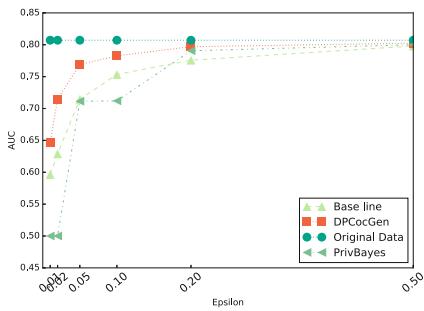


Fig. 5. Relationship prediction

7 Conclusion

This work presents an approach for the anonymization of microdata sets. The goal was to be able to produce synthetic data that preserve sufficient information to be used instead of the real data. Our approach involves combining differential privacy with synthetic data generation. We use co-clustering a data joint distribution estimation technique, in order to partition the data space in a differentially private manner. Then, we use the resulting partitions to generate synthetic individuals. We have shown that the synthetic data generated in this way retain the statistical properties of the raw data, thus using the synthetic data for various data mining tasks can be envisaged. We have also shown that our parameter-free approach outperforms other existing differentially private data release algorithms. We now plan to compare our approach to a previous work, that is being published, which is based on a group anonymization technique and we aim to articulate the discussion around the utility/protection trade-off.

References

1. <https://sourceforge.net/projects/privbayes>
2. Acs, G., Castelluccia, C., Chen, R.: Differentially private histogram publishing through lossy compression. In: 2012 IEEE 12th International Conference on Data Mining, pp. 1–10. IEEE (2012)
3. Blondel, V.D., Esch, M., Chan, C., Clérot, F., Deville, P., Huens, E., Morlot, F., Smoreda, Z., Ziemlicki, C.: Data for development: the D4D challenge on mobile phone data (2012). arXiv preprint [arXiv:1210.0137](https://arxiv.org/abs/1210.0137)
4. Boullé, M.: Data grid models for preparation and modeling in supervised learning. In: Hands-On Pattern Recognition: Challenges in Machine Learning, vol. 1, pp. 99–130 (2010)
5. 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). https://doi.org/10.1007/11787006_1
6. Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-79228-4_1
7. 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). https://doi.org/10.1007/11681878_14
8. Hartigan, J.A.: Direct clustering of a data matrix. *J. Am. Stat. Assoc.* **67**(337), 123–129 (1972)
9. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 19–30. ACM (2009)
10. Mohammed, N., Chen, R., Fung, B., Yu, P.S.: Differentially private data release for data mining. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493–501. ACM (2011)
11. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing, STOC 2007, pp. 75–84. ACM, New York (2007). <https://doi.org/10.1145/1250790.1250803>
12. Robert, C.: The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation. Springer, New York (2007)
13. Sweeney, L.: k-anonymity: A model for protecting privacy. *Int. J. Uncertainty Fuzziness Knowl. Based Syst.* **10**(05), 557–570 (2002)
14. Xiao, Y., Xiong, L., Fan, L., Goryczka, S.: Dpcube: differentially private histogram release through multidimensional partitioning (2012). arXiv preprint [arXiv:1202.5358](https://arxiv.org/abs/1202.5358)
15. Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G., Winslett, M.: Differentially private histogram publication. *VLDB J.* **22**(6), 797–822 (2013)
16. Zhang, J., Cormode, G., Procopiuc, C.M., Srivastava, D., Xiao, X.: Privbayes: Private data release via bayesian networks. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp. 1423–1434. ACM (2014)

Personal Analytics: An Individual and Collective Perspective

Evaluating the Impact of Friends in Predicting User's Availability in Online Social Networks

Andrea De Salve^{1,2(✉)}, Paolo Mori¹, and Laura Ricci²

¹ Institute of Informatics and Telematics, National Research Council, Pisa, Italy
pao.lo.mori@iit.cnr.it

² Department of Computer Science, University of Pisa, Pisa, Italy
{desalve,ricci}@di.unipi.it

Abstract. In recent years, Online Social Networks (OSNs) have changed the way people connect and interact with each other. Indeed, most people have registered an account on some popular OSNs (such as Facebook, or Google+) which is used to access the system at different times of the days, depending on their life and habits. In this context, understanding how users connect to the OSNs is of paramount importance for both the protection of their privacy and the OSN's provider (or third-party applications) that want to exploit this information. In this paper, we study the task of predicting the availability status (online/offline) of the OSNs' users by exploiting the availability information of their friends. The basic idea is to evaluate how the knowledge about availability status of friends can help in predicting the availability status of the center-users. For this purpose, we exploit several learning algorithms to find interesting relationships between the availability status of the users and those of their friends. The extensive validation of the results, by using a real Facebook dataset, indicates that the availability status of the users' friends can help in predicting whether the central user is online or offline.

Keywords: Personal behavior · Availability prediction
Online Social Networks

1 Introduction

Online Social Networks (OSNs) have attracted millions of users, that connect every day to the OSNs in order to share information with their friends or to directly interact with those who are online. The daily activities performed by each user produce a huge amount of private data that can be retrieved and exploited for different purposes, such as to predict the user's behavior. As for instance, the most part of current OSNs provide tools (such as the Chat Status) that can be used to obtain availability status (online/offline) of their users. In this scenario, a user u , who is friend of z , could access the availability status of the friends in common with z . Supposing that z wants to protect his availability

status from being disclosed to u , the availability information that u could collect about z' friends may be considered as a threat to the privacy of z , because u can exploit such knowledge to infer the availability status of z . Understanding how this information impact the predictability of the user's behavior is essential to protect the privacy of the OSNs' users. Unfortunately, the problem of predicting the availability status of a user z by exploiting the availability status of the friends of z has not been investigated for the privacy protection in a user-centric scenario (see Sect. 2). In addition, the problem of predicting the availability status has many real-world applications, like deciding what is the best time to send instant notification to a OSN's user and managing important problems in a distributed scenario (such as data availability and information diffusion [10]). The aim of this paper is to investigate the task of predicting availability status of OSN's users by exploiting a large real Facebook dataset, containing availability chat status of a set of users for 32 consecutive days. The data obtained from Facebook are used to train an extensive set of learning algorithms and to validate their ability in solving the task of predicting the availability status of the users. Using this sample of Facebook's users, we also evaluate the performance of each algorithm and we highlight some important properties and issues.

The rest of the paper is structured as follows: Sect. 2 provides an overview of the related works studying the temporal users' behavior in OSN while Sect. 3 presents the Facebook dataset used for the evaluation of our results, the general characteristics of the dataset (see Sect. 3.1), and the preparation of such dataset for the task of classification (see Sect. 3.2). Section 4 describes the classification algorithms used in our experiments while Sect. 5 validates their results by exploiting several quality measures. Finally, Sect. 6 reports the conclusions and discusses the future works.

2 Related Works

The study of temporal properties of OSNs' users have gained attention from researchers and several works have been proposed. In [17], authors propose to predict the availability status of a user's peer by combining results of different predictors. The first predictor labelled a users' peers based on their uptime status (namely, strongly/weakly offline, and strongly/weakly online). The second and the third predictors exploits Bruijn graph to represent the recent availability patterns of the users and to reduce the noise of variation on these patterns. Finally, the authors consider linear predictor that exploits the availability status of users in the last k days.

Authors of [4] demonstrated the existence of regular patterns in users' behavior. In addition, they implement a linear predictor which exploits the last 7 days of history to predict their online periods for the next week. However, the proposed approach is evaluated through an epidemic distributed protocol which chooses good partners by using the linear predictor.

Authors of [5] exploit a real dataset derived from MySpace to show that user's availability is correlated to both the time of the day and the presence of their friends on the platform.

Authors of [11] exploit probabilistic linear regression that is trained, by using different datasets, to minimize the Mean Squared Error. However, the authors do not use OSN dataset to train the model, but three different datasets derived from instant messaging and peer-to-peer applications.

Authors of [9] use a sample of Facebook users to investigate the relationships between the ego network structure of a user and the availability patterns of the friends in the ego network. In particular, the authors identified strong similarity (or temporal homophily) between the availability patterns of the users and their ego network's friends.

Authors of [2] analyzed four distinct OSNs derived from Orkut, MySpace, Hi5, and LinkedIn. In particular, they investigate the most recurrent actions performed by the users of the datasets and the types of contents shared with friends. Finally, another interesting analysis related to messages exchanged by the college students on Facebook is reported on [12]. The authors showed the presence of a weekly temporal pattern where users are clustered by similar patterns. They noticed the presence of a seasonal pattern in students' activities and showed that the weekend heavily impacts the typical users' patterns.

To the best of our knowledge, none of the previous works investigated the problem of predicting the availability status of the users by exploiting information derived from friends. In addition, the most part of related works adopt very simple models (such as linear predictor) to predict availability status of a user and they assume to know only the past availability status of the same user.

Table 1. General characteristics of the Facebook dataset

Name	Value
Start date	3 March 2015 - 10:05:00
End date	9 April 2015 - 12:55:00
Number of registered users	204
Total number of monitored users	66880
Number of time instants in a day T	288
Sampling frequency Δ	5 min

3 The Facebook Dataset

We focus on Facebook OSN because it enables its user to infer temporal information about the behaviour of their friends by exploiting the Facebook Chat Status. In particular, we developed a Facebook Application¹ which exploits the Facebook API to monitor the users registered to the application, by sampling periodically the chat status of such users and their friends. The Facebook Chat API allows to check if a user is logged on Facebook, regardless of the actions

¹ <https://www.facebook.com/SocialCircles-244719909045196/>.

he is doing during the session (such as browse contents or send messages). The temporal information collected from Facebook allow us to build session traces of users which indicate the presence or not of Facebook's users, at different time instants. Table 1 describes the characteristics of the dataset retrieved by our application. The Facebook application had been running for 32 days, i.e., from 9 March to 10 April 2015. During this time interval, we sampled the Facebook Chat status of all the registered users and their friends every 5 min. We decided to fix the sampling frequency (Δ) to 5 min because a shorter granularity was not possible for technical reasons related to the Facebook Graph API. In particular, our application was able to retrieve the following sets of information:

Friendship the set of friends of each registered user, and the friendship relations existing between them.

Online presence the availability status of the Facebook Chat of the registered users and their friends. The availability status can assume a limited set of value: 0 if user is offline, 1 if user is online.

Using this methodology we were able to access the availability status of 204 registered users and of their friends (for a total of 66.880 users). The set of registered users has the advantage of representing a very heterogeneous population: 114 males and 90 females, with age range of 20–79 with different education and background. In addition, the majority of the registered users have geographical location in Italy or in central Europe. For the sake of clarity, we assume a discrete time model, where the time t_i of a day d is represented by positive integers $i = 1, 2, \dots, T$ and the number of time instants T in a day is equal to 288 as it depends on the sampling frequency Δ , which is equal to 5 min.

3.1 Data Understanding

In this section we describe the general characteristics of our dataset. Due to technical reasons related to the memory space required by our application, the original dataset is a collection of records consisting of a pair $< t_i, u >$, where t_i is the time instant and u is the identity of a user that was found online at the corresponding time instant t_i . As a result, the dataset contains only temporal information about the users who are online for a total of 88,875,873 records. The boxplot of Fig. 1(a) provides a quick overview of the extreme characteristics of our dataset by showing the number of records collected for each user. The box corresponds to the interquartile range (IQR), covering the middle 50% of the data, and it goes from the lower quartile ($Q_1 = 25\%$) to the upper quartile ($Q_3 = 75\%$). We have collected more than 839 records for more than 50% of the users and the IQR ranges in the interval [264,1993]. The dotted line of the box indicates the $1.5 \cdot IQR$ and the data points outside this limit are drawn separately.

In addition, the plot indicates the presence of a higher number of user with only one record (i.e., 1.35% of the users are online only for 5 min during the whole

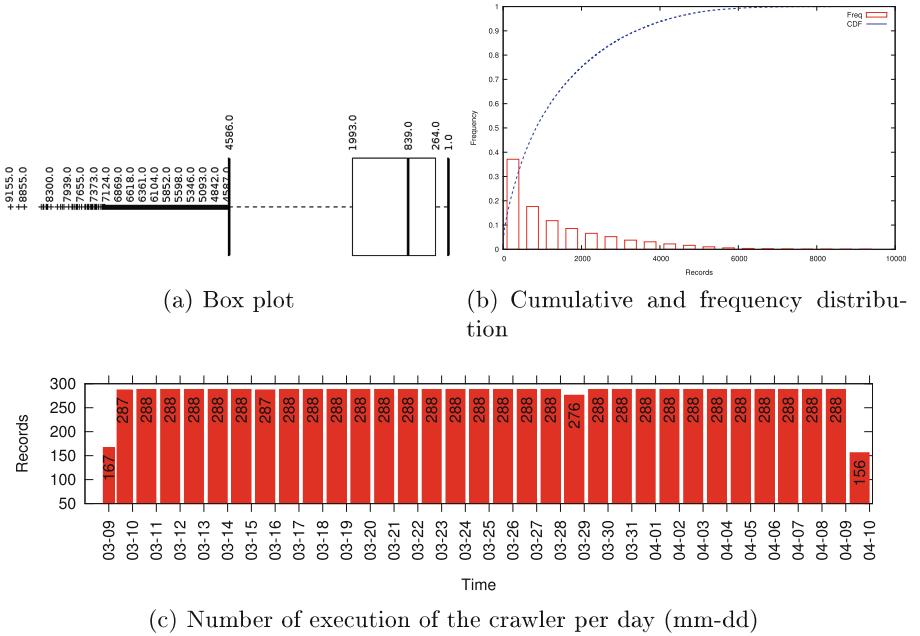


Fig. 1. The box plot (a) depicts statistical parameters of our dataset while graph (b) displays statistical distribution of the values. Finally, graph (c) shows the number of times in a day that our crawler has been successfully executed.

monitored period). The upper extremity of the box plot identifies the users who were online for the most part of the crawling period, having number of records in the range [4586, 9155]. The box plot of our sample came from the statistical distribution shown in Fig. 1(b), which depicts both the frequency distribution (Freq) and the Cumulative Distribution Function (CDF) of the number of records of the monitored users. The plot indicates that 70% of the users spend online short periods of time because we have collected less than 4000 records per user during the entire monitored period, while a small fraction of the users (about 10%) exposes more than 4000 records. In addition, we observed that there are no users who have been online for the entire monitoring period, i.e., having a number of records equals to 9250.

To achieve a deeper understanding of the collected data, we investigated the number of online/offline users over time. Figure 2 shows the total number of online/offline users, as well as, the fraction of online/offline users. The plot indicates the presence of a cyclic day/night pattern, which is confirmed also by results in [2, 9, 10, 12]. In particular, most of the users are connected during lunchtime and in the evening. In addition, we noticed that users who are offline, i.e., having availability status $S = 0$, are more than the users who are online (i.e., $S = 1$).

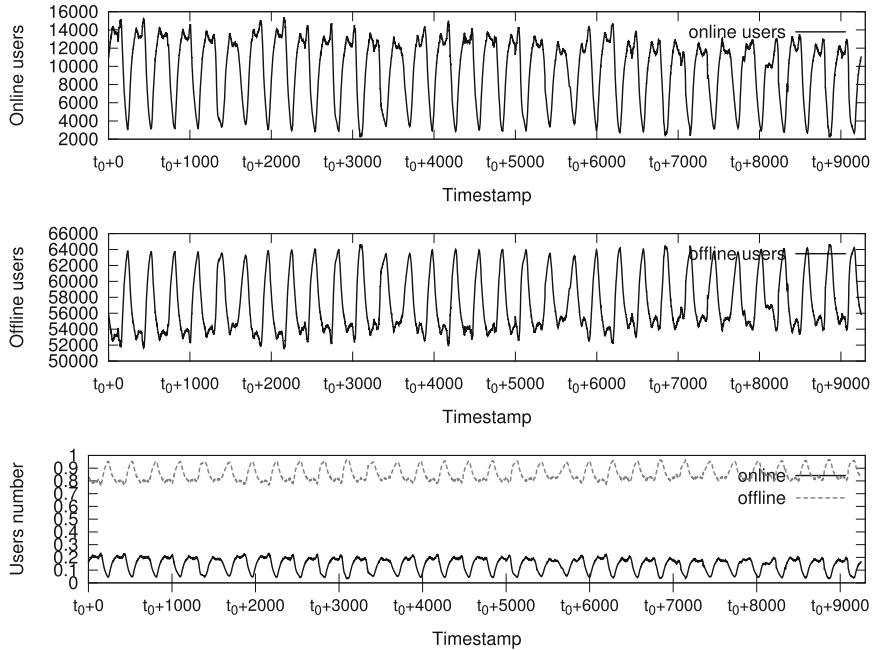


Fig. 2. The total number of online/offline users during the monitored period.

3.2 Data Preparation

In this section, we investigate the quality of our dataset and we describe how we transform the data in order to prepare them for the prediction task. Figure 1(c) shows, for each day of the monitored period, the number of times that our crawler has been executed without failure. Indeed, sometimes the Facebook API had blocked the execution of our application due to the excessive number of requests. We expected a total number of 288 executions per day (except for the first and the last day of the monitored period). The plot in Fig. 1(c) indicates that the execution of our crawler has failed once both on 10 and on 16 of April, while we have 12 failures on 29 of April. Since these missing values can lead to wrong data analysis results, we replaced these missing values with the availability status of users in the same time instants of the previous day (in the case of 16 and 29 of April) or of the consecutive day (in the case of 10 of April).

Since our dataset is a collection of records $\langle t_i, u \rangle$ which indicate only when user u is online, we perform a transformation phase on the original dataset and we construct new attributes that are necessary for the prediction task. Given a specific time instant t_i , we want to predict the availability status S of a user by exploiting the number of online user's friends and the number of offline user's friends at time t_i . The availability status of the user is a nominal target attribute S , whose value is equal to 1 if the user is online or 0 if the user

is offline in the corresponding time instant. For these reasons, we create for each time instant t_i and for each user u registered to our application a record $< onFriends, offFriends, S >$ which indicates the number of online/offline friends of u ($onFriends/offFriends$) along with the availability status of u at time instant t_i . Table 2 summarizes the set of attributes considered in each instance and the availability prediction problem consists in predicting the availability status S by exploiting such attributes: $onFriends, offFriends \Rightarrow S(0/1)$.

Table 2. Description of the attributed obtained from the dataset

Name	Type	Values/Format	Description
$onFriends$	Numeric	Integer	Number of online friends
$offFriends$	Numeric	Integer	Number of offline friends
S	Nominal	1 = online, 0 = offline	Availability status

4 Training and Classification Algorithms

We consider several supervised learning algorithms for the classification task and we split the Facebook dataset into two disjoint sets: the training data and test data. In our experiments the training data consist of the first 80% of the data while the remaining 20% is used for testing the learning algorithms. The two sets have empty intersection, i.e., they do not have common records. Table 3 summarizes the set of learning algorithms we considered in our experiments, as well as the input parameters. In the following, we briefly summarize these algorithms.

C4.5 Decision Tree Learner [18] is one of the most used methods for classification decision tree. The algorithm built a hierarchical decision tree which is used for classifying the availability status of a user, based on a number of attributes (i.e., the number of online/offline friends, in our case). The target attribute to predict are not used by the DT while the other attributes are used by the tree to routes an unclassified instance towards a leaf node, based on their values. In particular, each intermediate node checks if some condition is met on the value of the attributes and makes decisions about the next child node which must be considered, until a leaf node is reached. Each leaf node has a label that is used to classify the input instance as online or offline. The conditions on internal nodes of the DT are generated by splitting the domain of attributes in two partitions (i.e., using a binary split) and the Gini index is used as a quality measure to calculate such splitting point.

Rep Tree Learner algorithm is provided by Weka Mining Software [20] and it built a regular decision tree with reduced-error pruning by maximizing entropy values.

Random Decision Forests Learner extends the traditional DT Learner by constructing several decision trees [6]. The algorithm selects several random samples from the training data and a decision tree built from each sample. Finally, the resulting decision trees are combined with each other in order to produce a random forest that performs better than the original learners. In particular, the decision trees are joined by using a bootstrap aggregation procedure that assigns weights to average individual decision trees.

Functional Tree Learner [16] combines Logistic Regression [14] and Decision Tree [18] in order to classify instances. In particular, the algorithm built a decision tree where internal nodes are generated by splitting the domain of an attribute in two partitions. Instead, the leaves of the tree embed linear regressions that describe relationships between the number of online/offline friends and the availability status of a user. In particular, the logit function [14] is used to estimate the probability that user is online or offline based on the presence of some characteristics.

Naive Bayes Learner is based on Bayes's theorem [1] which is used to compute the most probable availability status of a user depending on the values of the other attributes. For this purpose, the conditional probability is used and the algorithm assumes that all the attributes are conditionally independent. The training data are used to fit the posterior probability and prior probability distribution while the Laplace corrector is used for estimating the model parameters based on the presence on categorical attribute values. Finally, the generated probabilistic model is used to predict the availability status S of a user by maximizing the conditional probability of S , given the number of online/offline friends of the user.

k-Nearest Neighbor is based on the nearest-neighbor algorithm [8], which classifies the availability status S of a user based on the availability status of the k most similar instances. The underlying algorithm exploits a KD-tree and the Euclidean distance for measuring the similarity between the instances. The prediction is computed by averaging the availability status of the k nearest neighbors. The number of neighbors to be considered is an input parameter and it is fixed to 20. In addition, since K-Nearest Neighbor is affected by the domain of numerical attributes, we decide to scale such domain to similar ranges, i.e., to the $[-1, 1]$.

Probabilistic Neural Network [13] based on the Dynamic Decay Adjustment (DDA) [3] allows to predict availability status of unclassified instances. The network consists of a 4 layers: (i) the input layer compares the input instances with the training data, (ii) the pattern layer computes the weighted product of the values and applies a non-linear transformation, (iii) the summation layer computes the sum of the values by considering both online and offline pattern, finally (iv) the output layer produces a binary output which correspond to the predicted availability status. The network is trained by exploiting Gaussian function that is tuned by two input parameters, *theta minus* and *theta plus* whose value is fixed by default to 0.2 and 0.4, respectively.

Table 3. Description of the learning algorithms used in our experiments

Name	Description
C4.5 Decision Tree Learner (DT)	No pruning, Gini index
Rep Tree Learner (RDT)	-
Random Decision Forests Learner (RDF)	#Trees = 10
Functional Tree Learner (FT)	-
Naive Bayes Learner	Laplace corrector = 0
k-Nearest Neighbor (k-NN)	k = 20
Probabilistic Neural Network (PNN)	theta minus = 0.2, theta plus = 0.4

5 Results Validation

We used the test data in order to evaluate the performance of the different learning algorithms. Since the most part of the Facebook's users connect to system only for short periods of time (see Sect. 3) we expect that the number of instances of the dataset having availability status equals to offline are higher than the number of instances with availability status S equals to online. In order to take into account this unbalance we performed an equal size sampling on the test set which allows to under-sample the instance having the most frequent availability status. We plan to investigate more sophisticated techniques for unbalance class distribution (such as [7]) and for the validation of the models (such as cross-validation) as future works.

5.1 Performance Measures

In order to compare the accuracy of the learning algorithms used in our experiments we calculated the following quality measures:

- A Confusion Matrix [19] is a table that represents: (i) the number of instances that have been correctly classified as online (true positive or TP) or as offline (true negative or TN), and (ii) the number of instances that have been classified as offline when they are online (false positive or FP), and (iii) the number of instances that have been classified as offline when they are online (false negative or FN).
- The Sensitivity measures the ability of the predictors to correctly classify the availability status of the users (i.e., $TP/(TP+FN)$).
- The Precision measures the ability of the predictors to correctly identify instances that are online (i.e., $TP/(TP+FP)$).
- The Specificity measures the ability of the predictors to correctly classify users having availability status equal to offline. It is obtained by calculating $TN/(TN+FP)$.
- The F-measure combines precision and sensitivity by using the harmonic mean and it is used to measure the accuracy of the test.

- The Accuracy is one of the most important measure because it indicates the ability of the predictor to classify instances that are both online or offline. In particular, it obtained by calculating $(TP+TN)/(TP+FP+TN+FN)$.
- The Cohen's kappa is an agreement index that measures the degree of accuracy and reliability of the classification task. Depending on the value of the Cohen's kappa, the index can be interpreted as [15]: (i) no agreement if $k \in [0, 0.20]$, (ii) fair agreement if $k \in [0.21, 0.4]$, (iii) moderate agreement if $k \in [0.41, 0.6]$, (iv) substantial agreement if $k \in [0.61, 0.8]$, and (v) perfect agreement if $k \in [0.81, 1]$.
- The Area Under Curve (AUC) is a measure that is derived from the Receiver Operating Characteristic Curve and it indicates the ability of the classifier in solving the problem of predicting the availability status of the users. The values of AUC belong to the interval $[0.5, 1]$ where 1 indicates perfect classification without errors while 0.5 corresponds to random classification.

Table 4. Confusion matrix of the predictors.

Name	Confusion matrix			
	TP	FP	TN	FN
C4.5 Decision Tree Learner	25919	5621	68620	48322
Rep Tree Learner	23792	5004	69237	50449
Random Decision Forests Learner	25765	5638	68603	48476
Functional Tree Learner	23928	5159	69082	50313
Naive Bayes Learner	7710	4212	70029	66531
k-Nearest Neighbor	25575	5694	68547	48666
Probabilistic Neural Network	24328	4966	69275	49913

5.2 Results Evaluation

The Table 4 shows the confusion matrix, indicating the absolute number of instances which have been correctly classified by each algorithm, as well as the number of incorrectly classified instances. From the columns TP and TN we can easily derive the total number of correct predictions made by each algorithm, as well as the total number of incorrect predictions (i.e., the columns FP and FN). The reader can notice how the total number of correct predictions of the C4.5 Decision Tree Learner outperforms the others. In addition, a significant number of correct predictions is also achieved by the Random Decision Forest Learner, the K-Nearest Neighbor, and the Probabilistic Neural Network. Instead, the Naive Bayes Learner is the worst in terms of the number of incorrect predictions.

The quality measures used for evaluating the models can be calculated from the confusion matrix and they are shown in Table 5 for the sake of clarity. As we

expected, the sensitivity of the classifier in predicting availability status of online is not very high and it does not exceed 0.35. Indeed, the higher number of records having availability status equals to offline heavily affects the class distribution and reduce the number of instances which are useful for the prediction of the online users. For this reason, the algorithms may fail in predicting the availability status of the users when they are online. Indeed, the specificity measure is very higher for almost all the predictors and it clearly indicates the ability of the predictor in identifying the availability status of the offline users.

However, the predictors show to have higher precision, indicating that the most part of users having availability status equals to online are correctly identified by the predictors. The F-measure summarizes the performance of each predictor for the online case and it shows that predictors based on Decision Tree have the best performance. The last step in our analysis consists in evaluating the accuracy of the predictors. Table 6 reports the Accuracy, the Cohen's kappa and the AUC obtained by each predictor. The most part of the predictors have an Accuracy value that does not exceed 0.65. The agreement index (Cohen's kappa) is also fine because it is higher than 0.20 (except for the predictor based on Naive Bayes Learner). Finally, the AUC value clearly indicates the C4.5 Decision Tree Learner, Random Decision Forests Learner, K-Nearest Neighbor, and Probabilistic Neural Networks are the most promising algorithms in solving the

Table 5. Performance of the predictors

Name	Sensitivity	Precision	Specificity	F-measure
C4.5 Decision Tree Learner	0.349	0.822	0.924	0.490
Rep Tree Learner	0.321	0.826	0.932	0.462
Random Decision Forests Learner	0.347	0.821	0.924	0.488
Functional Tree Learner	0.322	0.823	0.931	0.463
Naive Bayes Learner	0.104	0.647	0.943	0.180
k-Nearest Neighbor	0.345	0.818	0.923	0.485
Probabilistic Neural Network	0.328	0.831	0.933	0.470

Table 6. Accuracy of the predictors

Name	Accuracy	Cohen's Kappa	AUC
C4.5 Decision Tree Learner	0.637	0.273	0.769
Rep Tree Learner	0.627	0.253	0.763
Random Decision Forests Learner	0.636	0.271	0.803
Functional Tree Learner	0.626	0.253	0.645
Naive Bayes Learner	0.524	0.047	0.608
k-Nearest Neighbor	0.634	0.268	0.802
Probabilistic Neural Network	0.630	0.261	0.801

task of availability prediction of the OSNs' users because they enable an attacker to infer the availability status of a user for at least 60% of the time.

6 Conclusion and Future Works

In this paper, we uncovered a number of interesting results related to the problem of predicting the availability status (online/offline) of OSNs' users. In particular, we showed that the availability status of an individual is partly affected by those of their friends and we found that Decision Tree Learner and Random Decision Forest Learner have the best accuracy in predicting the availability status. In addition, we observed that k-Nearest Neighbor and Probabilistic Neural Network are suitable models for predicting the user's availability.

As future work, we would like to investigate different configuration parameters of the considered learning algorithms. In addition, we plan to deal the class's unbalance problem by exploiting advanced techniques, such as Synthetic Minority Over-sampling [7]. Another interesting aspects is to boost the performance of the predictors by considering other attributes useful for predicting the availability status of the users, such as the timestamp and sessions length. Finally, we plan to exploit association analysis to identify relationships between the availability status of users and the identities of the users' friends who are online/offline.

References

1. Baron, M.: Probability and Statistics for Computer Scientists. CRC Press, New York (2013)
2. Benevenuto, F., Rodrigues, T., Cha, M., Almeida, V.: Characterizing user behavior in online social networks. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, pp. 49–62. ACM (2009)
3. Berthold, M.R., Diamond, J.: Constructive training of probabilistic neural networks. Neurocomputing **19**(1), 167–183 (1998)
4. Blond, S.L., Fessant, F.L., Merrer, E.L.: Choosing partners based on availability in P2P networks. ACM Trans. Auton. Adapt. Syst. (TAAS) **7**(2), 25 (2012)
5. Boutet, A., Kermarrec, A.M., Le Merrer, E., Van Kempen, A.: On the impact of users availability in OSNS. In: Proceedings of the Fifth Workshop on Social Network Systems, p. 4. ACM (2012)
6. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
8. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Tran. Inf. Theory **13**(1), 21–27 (1967)
9. De Salve, A., Dondio, M., Guidi, B., Ricci, L.: The impact of user's availability on on-line ego networks: a facebook analysis. Comput. Commun. **73**, 211–218 (2016)
10. De Salve, A., Guidi, B., Mori, P., Ricci, L., Ambriola, V.: Privacy and temporal aware allocation of data in decentralized online social networks. In: Au, M.H.A., Castiglione, A., Choo, K.-K.R., Palmieri, F., Li, K.-C. (eds.) GPC 2017. LNCS, vol. 10232, pp. 237–251. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57186-7_19

11. Dell'Amico, M., Michiardi, P., Roudier, Y.: Back to the future: on predicting user uptime. CoRR abs/1010.0626 (2010). <http://arxiv.org/abs/1010.0626>
12. Golder, S.A., Wilkinson, D.M., Huberman, B.A.: Rhythms of social interaction: messaging within a massive online network. *Commun. Technol.* **2007**, 41–66 (2007)
13. Haykin, S.S.: Neural Networks and Learning Machines, vol. 3. Pearson, Upper Saddle River (2009)
14. Hilbe, J.M.: Logistic regression. In: Lovric, M. (ed.) International Encyclopedia of Statistical Science, pp. 755–758. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-04898-2_344
15. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. *Biometrics* **33**(1), 159–174 (1977)
16. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. *Mach. Learn.* **59**(1–2), 161–205 (2005)
17. Mickens, J.W., Noble, B.D.: Exploiting availability prediction in distributed systems. *Ann Arbor* **1001**, 48103 (2006)
18. Quinlan, J.R.: C4.5: Programs for Machine Learning. Elsevier, San Francisco (2014)
19. Stehman, S.V.: Selecting and interpreting measures of thematic classification accuracy. *Remote Sens. Environ.* **62**(1), 77–89 (1997)
20. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2016)

Movement Behaviour Recognition for Water Activities

Mirco Nanni, Roberto Trasarti^(✉), and Fosca Giannotti

ISTI CNR - KDD Lab, Pisa, Italy

{mirco.nanni,roberto.trasarti,fosca.giannotti}@isti.cnr.it

Abstract. This work describes an analysis process for the movement traces of users during water activities. The data is collected by a mobile phone app that the Navionics company developed to provide to its users sea maps and navigation services. The final objective of the project is to recognize the prevalent activity types of the users (fishing, sailing, cruising, canoeing), in order to personalize services and advertising.

1 Introduction

Mobility data analysis has been a very hot topic in the last decade due to the wide diffusion of localization technologies – especially GPS devices on board of vehicles and location-based services for smartphones – as well as the large interest in understanding human mobility in urban settings, e.g. for smart-city applications.

Quoting one of its mottos, the main activity of the Navionics company¹ “starts where the roads end”, providing electronic charting to the ease and safety of navigation to around one million users worldwide. Nowadays, that is accomplished through a GPS-enabled smartphone app that also allows to collect the users’ tracks (under her consent) to improve the service.

This paper focuses on a specific analytical task on such data: inferring the main water activities of interest for the user, in particular distinguishing those interested in fishing, from those keen on sailing or canoeing, and those spending most of the time simply cruising around. The impact on the company is at least two-fold: on one hand that would lead to better customizing the navigation and mapping services to the user (e.g. highlighting on the map popular fishing spots only to fishermen); on the other hand, it would allow more effective target marketing initiatives.

The task presents several challenges, mainly due to the large heterogeneity of the users’ behaviours, due both to individual attitudes (a sailor obviously behaves very differently from a fisherman, but there are also several different ways to sailing, such as competitive vs. enjoyable, beginner vs. expert, etc.) and the geographical context (fishing in Australia might be very different from the US coasts or the Great lakes). That requires to abstract, to some extent, from

¹ www.navionics.com.

the geography and take into consideration the existence of several different (and largely unknown a priori) behaviour profiles for the same activity type.

The paper will describe the main phases of the process, highlighting the challenges faced. After discussing the state-of-art, we will introduce the data sources adopted and the preprocessing phases, which were the most time consuming steps. Next, the clustering-based selection of typical behaviors and their labeling based on expert's input is described, followed by the construction of a decision tree-based classifier. The paper finishes with a study the results obtained with the application of the model on the whole dataset.

2 Related Work

Several works exist in literature dealing with human mobility, trying to model and recognize users' daily activities either by looking at their movement trajectories [5, 6] or at smartphone sensors, e.g. accelerometers, etc. [7].

Moving to boats and water activity, much less work is available. Most of it deals with AIS (automatic identification system) [1], i.e. a tracking system used for collision avoidance on ships and by vessel traffic services. Typical works try to study specific locations with issues [8] (crowded harbors, frequent maneuvers, etc.) or to detect some kinds of events [2] (e.g. collision risks). Also, some works are devoted to recognize very specific patterns, e.g. fishing in some specific (e.g. protected) areas [3].

The task we focus on is related to the analysis of individuals and their habits, which is poorly discussed in literature and there exist basically no suitable proposals for tackling the problem.

3 Data Preparation

3.1 Input Data Sources

The analysis process is based on two types of data sources: the tracks recorded by users of the Navionics app and various geographical information of the areas where the users moved.

Tracks data record the trips performed by a user when the Navionics app is activated. Tracks are basically sequences of GPS points that allow to reconstruct where and when the trip took place.

Various sources of **geography** data provide information about the borders of *land* (used to remove points outside water, deemed not interesting for this project), a spatial description of *coastline* (used to asses the distance of the user from the nearest coast) and borders delimiting different *types of water*, the type of *bottom* in each point in water (represented as polygons of homogeneous bottom type), a measure of *water depth* at each geographical point.

Finally, additional information about **attractiveness of places** was available, in particular a (large but partial) database of *wrecks* localized by Navionics users, and a *popularity heatmap* of each geographical area, inferred from the track data as number of users that visited each place.

3.2 Pre-processing

After cleaning the raw track data from points outside water, which are not interesting for our purposes and were most likely collected by mistake (for instance the user forgot to switch the app off), we analyze the user's trajectories to split them into subtrajectories, called *components*, that can belong to two categories: *movements* and *stops*.

The key operation is to define when a boat is still and when it is moving. Indeed, both the floating of a boat when it is not really moving (e.g. when anchored) and the fluctuations of GPS localization can create an apparent movement, therefore a total stop usually does not exist. For this reason, we operationally defined the status of a boat at a given moment as a stop if it remains very close to its initial position (w.r.t. a spatial threshold) for a significant amount of time (w.r.t. a minimum duration threshold). Both thresholds are parameters provided by the analyst. In the experiments shown later in the paper, they were empirically set to 50 m and 10 min. Complementary, all the points that do not belong to a stop period represent movements.

Each single point of a component was characterized by several features derived either from the local movement performed (estimate speed and angle) or from the area around its geographical position. In particular, in this project we selected the following basic features: *speed*; *angle*, which represents the turn angle described by three consecutive points; *distance from the coast*; *sea bottom type* (sand, rock, etc.); *depth*; *water type* (salted, fresh); presence of *wrecks*; *sea bottom steepness*, estimated from depth information.

Based on them, component-level features were extracted. Several of them are obtained through simple aggregation, such as median value, inter-quartile values and range, or taken from the first point of the component, such as the initial timestamp and latitude. Others required more complex processes. In particular, we shortly describe here *acceleration periods* and *noodle shapes*, while other examples are omitted for lack of space. Detecting the **accelerations** of a boat in a stable way requires to evaluate the speed over a relatively long period of time. Here, the speed at each point of a track is compared against previous ones, in particular those that are more recent than a given temporal threshold (fixed to 2 min in our experiments). If the present speed is higher than the minimum previous speed in such interval by more than a fixed threshold (fixed to a very small 0.55 m/s) and more than a fixed percentage (fixed to 20%), then the current point is considered an acceleration point. **Noodle shapes** represent wandering movements around a fixed area, for instance waiting for the fish to bite. In terms of trajectories, that results in forming very entangled shapes. The heuristics we adopted consists in computing the ratio between the length of a track segment and the actual distance traveled by the boat – as the crow flies. Entangled trajectories tend to have a high ratio, and we experimentally set a threshold for this parameter to 2.5, i.e. the track between two endpoints is 2.5 times larger than their spatial distance. Clearly, noodles might appear only in small segments on a component, and therefore we need to look for appropriate intervals within the component. We scan all the points of the component, and

for each of them look ahead for the first point that gets farther than 2 km from the starting point. This interval is then evaluated as described above, and if a noodle results, it is recorded and the scan proceeds from the endpoint (the noodle segment is skipped). Otherwise, the scan goes on. Notice that the 2 Km threshold (a input parameter) defines the granularity of noodles, i.e. the geographical scale at which look for noodle shapes. Small values (e.g. 50 m) would capture very local wanderings, rather similar to stops. Large values (e.g. 50 km) would miss fishing spots, while capturing larger exploration areas.

4 Modelling Step 1: Components Clustering

Our objective, here, is to build a catalog of representative user behaviours by grouping the components into homogeneous clusters, i.e. the components in a cluster are similar to each other.

The results will be used later for two purposes: first, clusters will be used to extract a stratified sample of data to be used as training set – after that domain experts analyzed and labeled them appropriately; second, the cluster representatives will be a key part of the activity recognition process, since it will indeed start with associating each component of the new tracks (to classify) to its most similar cluster.

4.1 Correlation-Based Features Selection

In order to identify possible issues of hidden redundancy among features, which might affect the creation of clusters, we computed all pair-to-pair Pearson's correlation coefficients, and used the corresponding correlation graph to select a subset of redundant features to remove. Figure 1 reports the strongest relations discovered (cyan = between 75% and 90%, black = above 90%), either positive or negative, and the features discarded (red crosses).

4.2 K-Means Clustering

Among the several alternative clustering methods available in literature, we chose to adopt the simple K-means algorithm, which is efficient and provides results that are easy to interpret – both features were very important for this phase of the project.

K-means [4] seeks K clusters that contain objects as much homogeneous as possible, in such a way that all objects in a cluster are usually rather similar to each other, and rather different from those in other clusters. Similarity between objects is computed as Euclidean distance between the feature vectors for the objects. The algorithm also outputs a centroid for each cluster, i.e. a synthetic objects whose features averaging those of all the objects in the cluster.

The value of parameter K has to be provided by the analyst. Since the preliminary exploratory experimentation we performed did not let emerge any critical

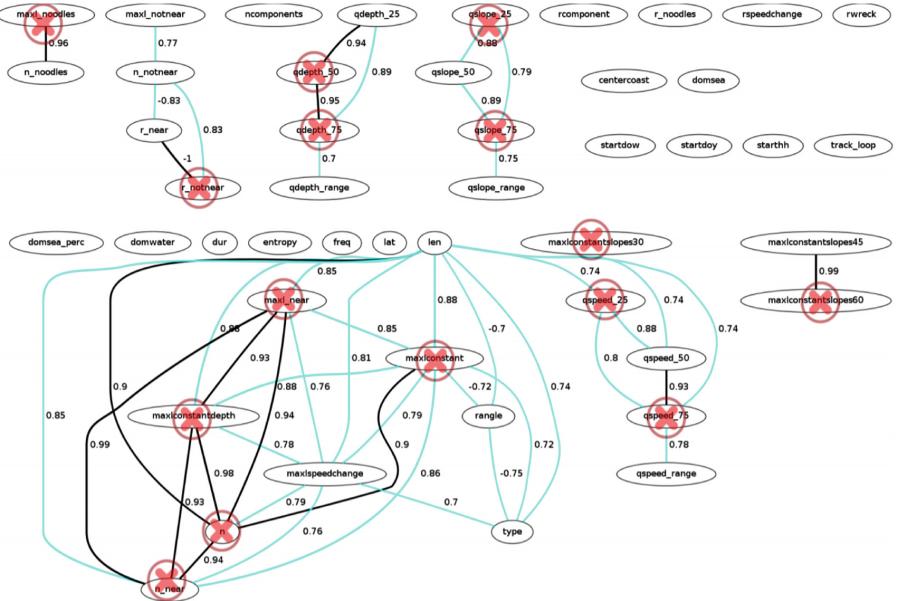


Fig. 1. Correlation graph computed over the component features. Features signed with a cross were later removed to keep only relevant and non-redundant information for the clustering step. (Color figure online)

value of K , we chose to set $K = 100$ as a trade-off between our two main objectives: (i) have enough clusters to capture the different possible users' behaviours, and (ii) keep the number of clusters small enough to make it feasible, for a domain expert, to observe and label a reasonable number of sample components that belong to each cluster.

Experimental results on our data show a set of rather well-balanced clusters (see size distribution in Fig. 2), with very few small ones and no *giant component*.

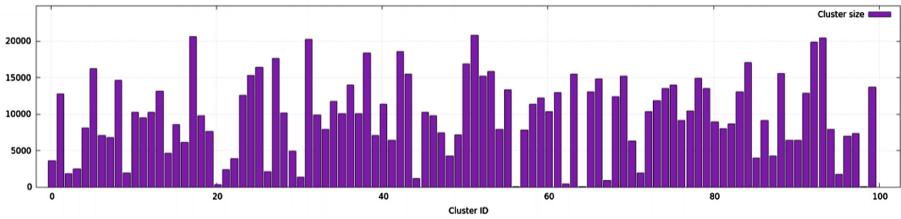


Fig. 2. Size of the 100 clusters obtained.

5 Modelling Step 2: Component Classification

The clusters found in the previous phase of the process represent significant behaviours of the users, yet it is still unknown to us exactly what type of activity corresponds to each cluster. This step involves the knowledge of a domain expert, which will be exploited to build the final classification model.

5.1 Training Set Construction and Division in to World Areas

Our objective, here, is to identify a reasonable set of tracks (and corresponding components) to be labeled by domain experts. In order to characterize our clusters, we start selecting a fixed number of components for each cluster (in our case set to 5), in particular focusing on components that are close to the representative centroid. That ensures both their representativeness and the fact to consider, in a limited amount, the variability of behaviours within each cluster.

Also, since similar behaviours might have different meaning or different weights in different areas of the world, the whole learning process will be differentiated over various sections of the world. In particular, the world has been divided into 6 macro-areas: USA East coast (USE), USA West coast (USW), Australia (Aus), Mediterranean sea (Med), Scandinavia (Scand), and United Kingdom (UK).

For each geographical area and each cluster, then, the 5 components closest to cluster centroid are selected, together with the complete track they belong to. This yields 6 different training sets to be labeled by the domain experts, each composed of around 500 tracks and their associated components. Our experiments involved Navionics experts of the areas above, who manually analyzed the track data and assigned an activity label to each component and track.

5.2 Clusters Labeling

The information provided by the experts over the sample of components they labeled is propagated to the clusters they belong to in the following way.

For each cluster we compute a probability distribution over the set of possible activities, obtained mixing the two levels of information provided by the experts:

- component-level: the number of components with that specific activity label;
- track-level: the number of components that belong to a track having that activity as overall label.

For each activity, the two counts above are merged through a weighted sum. The optimal setting obtained empirically consists in a weight of 0.85 for track-level labels, and 0.15 for component-level ones. That means that the label of the overall track is the most important information to recognize activities, yet the labels of the components still contribute significantly to it.

The reason for keeping a distribution of probability for a cluster, instead of just picking one representative activity, is that often the secondary activities

have a significant probability, and therefore discarding them would artificially polarize clusters and loose information. This is supported by Fig. 3, which shows for each cluster the dominant activity and its probability.

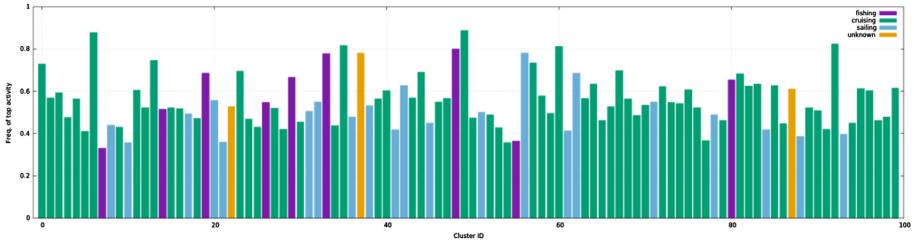


Fig. 3. Most frequent activity of each cluster and its frequency. (Color figure online)

From the plot it is possible to see that cruising dominates in several clusters, meaning that it is an activity capturing a large variety of behaviours, whereas fishing and sailing tend to have a lesser variability. Also, we can see that while some clusters are almost pure (they contain close to 100% of a single activity), most of them are quite mixed, leaving a lot of space (in most cases around 50%) for the secondary activities. Finally, a few clusters (orange in the figure) mainly contain components of unknown activity. No cluster was dominated by the canoeing activity, which was expected due to its low presence in the training set provided by the domain experts.

The process has been performed separately for each geographical region, thus yielding a set of labeled clusters for USE, one for USW, etc.

5.3 Clustering-Based Meta-features for Tracks

The labeled clusters obtained joining clustering and expert labeling provides a way to map any component to a set of potential activities performed. This operation can be performed by simply associating the component to the closest cluster, and then take its probability distribution.

Since our final objective is to learn to classify tracks, and not single components, we aggregate the distributions of all the components into a single track-level distribution. This is obtained by just averaging the probability of each activity label across all the components.

This final distribution might already be considered a classification of the track, the idea being to associate it to the predominant activity. Yet, in order to make our model flexible and to exploit all the information sources we have, we use this first “pseudo-”classification as an input for a later machine learning step that integrates also other information. These inputs, that we call “meta-features”, include the following data:

- predominant *activity* label and probability distribution over all activities;
- predominant *boat type* (another information provided by domain experts labeling) and probability distribution over all boat types;
- predominant *water type* (salted, fresh, salt lakes) and probability distribution over all water types;
- predominant *area type* (also these were provided by domain experts: near-shore, off-shore, intra-coastal) and probability distribution over all areas.

5.4 Training Set: Predictive Features and Target of Classification

The final classification model is built over a set of predictive features composed of all the meta-features described in the previous sections, plus some additional attributes derived from them. These extra attributes have the purpose of providing the classification models a list of explicit comparative measures of (the probabilities of) some key activities that might otherwise be not detected automatically. This list includes:

- $P(\text{fishing}) - P(\text{cruising})$: this allows to understand how much predominant is one activity over the other.
- $P(\text{fishing}) - P(\text{sailing})$: same as above.
- $P(\text{anchored}) + P(\text{docked}) + P(\text{drifting})$: this sums up a set of activities that were deemed significantly distinct by the domain experts, yet they all imply a stationary behaviour, virtually equivalent to a stop.

5.5 Classification Model

In this work we adopted the well-known and widely adopted Decision Tree model, consisting of a tree structure that starts from a root and follows some specific sequence of branches according to the values of some specific attributes.

A big advantage of using decision trees over other solutions is the readability of the model, which allows to check its sensibleness and better debug it.

The algorithm adopted for inferring a decision tree out of the training set is the standard C4.5 algorithm, which builds the tree in a incremental way, starting from a trivial root-only tree and expanding it by adding splitting nodes (basically the tests that create branches in the tree, such as is $P(\text{fishing}) > 0.6?$). Also, in order to evaluate some performance indicators of the models constructed, we adopted a cross-validation approach, where (a large) part of the training set is used to build the model, while another (smaller) part is used to check how well the predictions of the model fit the actual labels.

The two main parameters of the algorithm, namely the minimum number of objects in each leaf and the confidence factor of leaves (i.e. how much should the dominant label of a leaf predominate on the others) have been set by a simple grid search, choosing the combination of values that maximize the accuracy computed through cross-validation.

5.6 Performances of the Models

Figure 4 summarizes some characteristics of the models obtained on each area of the world.

Area	min_leaf	missed classes	Accuracy
USW	7	(canoeing)	70.21%
USE	7	-	71.35%
UK	13	?, (canoeing)	66.11%
Scand	5	?, (canoeing)	65.81%
Med	19	?, (canoeing)	64.53%
AU	11	sailing, ?, (canoeing)	69.01%

Fig. 4. Summary table of the decision trees characteristics.

The table shows that the accuracy ranges between 64% and 71%, which looks reasonably good considering the high number of classes involved (five, including the “?” class). Also, in most cases the models were not able to capture the *canoeing* class simply because the domain experts did not provide a sufficient number of cases. In the Australia dataset, instead, *sailing* was not captured, although present in the experts’ labeled data. *USE* was the richest dataset, and indeed the model performs best.

6 Using the Classifier

The ultimate objective of the project is to assign to all tracks of all users an activity label, exploiting the knowledge encoded in the classification models built in the previous sections. Then, through aggregation of the inferred labels we can assign an overall category to each user, based on what are the main activities (s)he performed.

6.1 Labeling New Tracks

Given a dataset of new, unlabeled tracks, in order to assign them to an activity, we basically need to follow the same preparation process performed during the model construction.

Figure 5(right) shows how the tracks were distributed across the different categories. That is compared against the distribution of components provided by the domain experts in the training phase, described in Sect. 5.1 (Fig. 5(left)). As we can observe, the two distributions are rather similar. The main differences include the fact that *cruising* looks more present now in the USE, USW and Mediterranean areas, whereas it dropped dramatically in UK and Scandinavia. Also, as already noticed in previous sections, *sailing* completely disappeared in Australia, since its model did not capture that category.

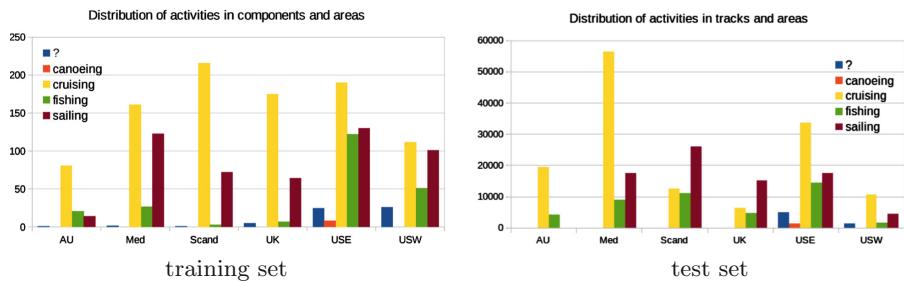


Fig. 5. Activity distribution: (i) training set (number of components assigned by domain experts to each activity); (ii) test set (number of tracks classified to each activity).

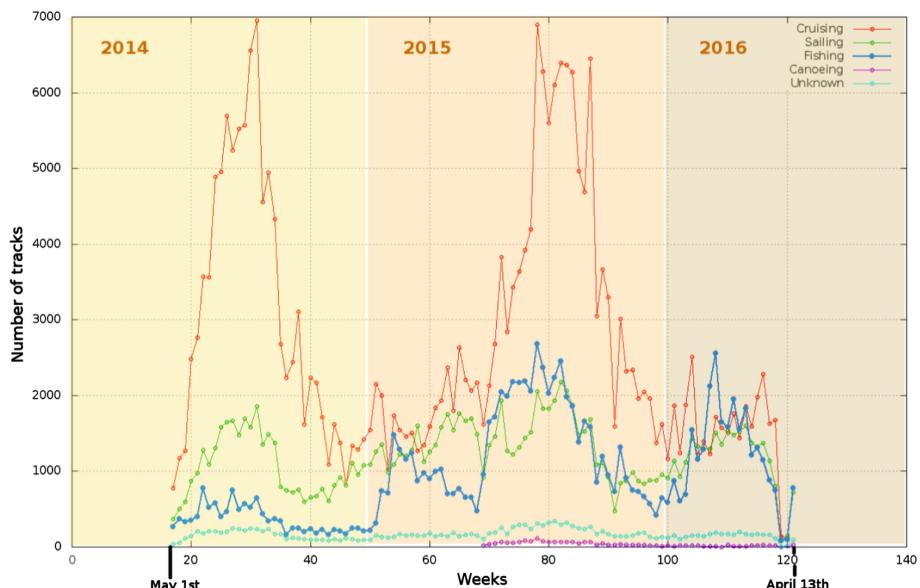


Fig. 6. Temporal distribution of activities along 2 years (mid-2014 to mid-2016).

An interesting view on the data can be obtained plotting the temporal distribution of the activities along the whole duration of the data we had access to, i.e. from May 2014 to April 2016, as shown below (Fig. 6).

In addition to the usual seasonal behaviours – overall increase of all activities in the summer months – we can observe that fishing increased sensibly its presence in the data during the last year. Possible causes might be an increased number of fishermen among Navionics users, or an increased propensity among fishermen to share their tracks, or a combination of the two.

6.2 Labeling Users

The labels assigned to each single track can be simply aggregated (counted) to infer the distribution of activities for each user. The next step, then, consists in selecting the activity – or activities – that represent(s) the user best. After some trials and evaluation of the results with the domain experts, the following approach was decided:

- If the user has a percentage of fishing tracks larger than 30%, we label the user as “fisherman”, since at the present fishing is considered a strategic segment of customers.
- Otherwise, the label with the largest percentage is selected, with no minimum thresholds.

6.3 Sample Statistics of Results

Figure 7 shows the distribution of users’ activities for each area. In the left plot we assign to each user the predominant activity, while in the second one we adopt the incentive mechanism for “fishing” mentioned above. Figure 8 summarizes the same information in tabular form and percent distribution.

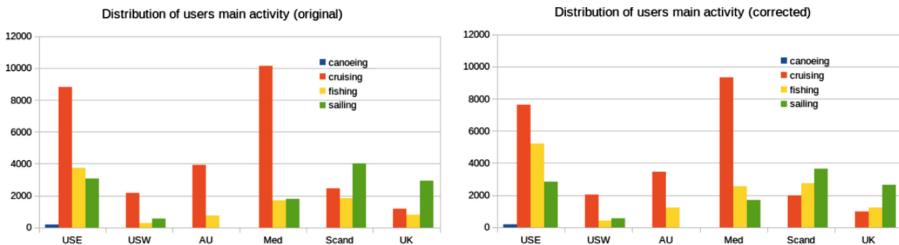


Fig. 7. Distribution of users’activity (left) and effects of fishing incentives (right).

	Original					
	USE	USW	AU	Med	Scand	UK
canoeing	1.29%	0.00%	0.00%	0.00%	0.00%	0.00%
cruising	55.66%	72.42%	83.65%	74.20%	29.36%	23.87%
fishing	23.70%	8.50%	16.35%	12.48%	22.25%	16.11%
sailing	19.36%	19.08%	0.00%	13.32%	48.38%	60.01%

	Corrected					
	USE	USW	AU	Med	Scand	UK
canoeing	1.15%	0.00%	0.00%	0.00%	0.00%	0.00%
cruising	48.09%	67.38%	73.95%	68.65%	23.70%	20.20%
fishing	32.84%	14.52%	26.05%	18.88%	32.72%	25.07%
sailing	17.93%	18.10%	0.00%	12.48%	43.58%	54.73%

Fig. 8. Percentage distribution of activities without and with fishing incentives.

As we can see, “cruising” remains the top activity in most areas, exceptions being Scandinavia and UK. Also, the small thresholding mechanism applied to fishermen yielded significant increases to their distribution.

7 Conclusions and Future Work

We proposed an analytical process to build a classification model able to infer the main water activities of an user having his recent traces. In particular the considered activities are: fishing, sailing, canoeing, and cruising. The methodology is presented and evaluated on a real use case with more than two years of worldwide data. Several challenges in managing such data are solved with general solutions making the process abstract and flexible. The final results can be used by the company to customize the navigation and mapping services to the user as well as propose a more effective target marketing initiatives. In the future we want to focus in obtaining better expert data in order to test the proposed methodology with an higher level of confidence in the different world areas. Moreover the original activities can be the initial nodes of an ontology of more specific behaviors which can be pushed in the system.

References

1. Claramunt, C., Ray, C., Camossi, E., Jousselme, A.-L., Hadzagic, M., Andrienko, G.L., Andrienko, N.V., Theodoridis, Y., Vouros, G.A., Salmon, L.: Maritime data integration and analysis: recent progress and research challenges. In: Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, 21–24 March 2017, pp. 192–197 (2017)
2. Napoli, A., Gallen, R., Bouju, A., Ray, C., Iphar, C.: DeAIS project: detection of AIS spoofing and resulting risks
3. de Souza, E.N., Boerder, K., Matwin, S., Worm, B.: Improving fishing pattern detection from satellite AIS using data mining and machine learning. *Plos One* **11**(7), 1–20 (2016)
4. Jin, X., Han, J.: K-means clustering. In: Sammut, C., Webb, G.I. (eds.) *Encyclopedia of Machine Learning*, pp. 563–564. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-30164-8_425
5. Liao, L., Fox, D., Kautz, H.: Extracting places and activities from GPS traces using hierarchical conditional random fields. *Int. J. Robot. Res.* **26**(1), 119–134 (2007)
6. Rinzivillo, S., Gabrielli, L., Nanni, M., Pappalardo, L., Pedreschi, D., Giannotti, F.: The purpose of motion: learning activities from individual mobility networks. In: International Conference on Data Science and Advanced Analytics, DSAA 2014, Shanghai, China, 30 October–1 November 2014, pp. 312–318 (2014)
7. Ronao, C.A., Cho, S.-B.: Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Syst. Appl.* **59**, 235–244 (2016)
8. Zhu, F.: Abnormal vessel trajectories detection in a port area based on AIS data. In: ICTE 2015, pp. 2043–2049 (2015)

Automatic Recognition of Public Transport Trips from Mobile Device Sensor Data and Transport Infrastructure Information

Mikko Rinne^(✉), Mehrdad Bagheri, Tuukka Tolvanen, and Jaakko Hollmén^(✉)

Department of Computer Science, Aalto University, PO Box 15400,
00076 Aalto, Espoo, Finland

mikko.rinne@iki.fi, {mehrdad.bagheri, tuukka.tolvanen, jaakko.hollmen}@aalto.fi

Abstract. Automatic detection of public transport (PT) usage has important applications for intelligent transport systems. It is crucial for understanding the commuting habits of passengers at large and over longer periods of time. It also enables compilation of door-to-door trip chains, which in turn can assist public transport providers in improved optimisation of their transport networks. In addition, predictions of future trips based on past activities can be used to assist passengers with targeted information. This article documents a dataset compiled from a day of active commuting by a small group of people using different means of PT in the Helsinki region. Mobility data was collected by two means: (a) manually written details of each PT trip during the day, and (b) measurements using sensors of travellers' mobile devices. The manual log is used to cross-check and verify the results derived from automatic measurements. The mobile client application used for our data collection provides a fully automated measurement service and implements a set of algorithms for decreasing battery consumption. The live locations of some of the public transport vehicles in the region were made available by the local transport provider and sampled with a 30-s interval. The stopping times of local trains at stations during the day were retrieved from the railway operator. The static timetable information of all the PT vehicles operating in the area is made available by the transport provider, and linked to our dataset. The challenge is to correctly detect as many manually logged trips as possible by using the automatically collected data. This paper includes an analysis of challenges due to missing or partially sampled information, and initial results from automatic recognition using a set of algorithms comparing measured trips with both live vehicle locations and static timetables. Improvement of correct recognitions is left as an ongoing challenge.

1 Introduction

Automatic detection of the door-to-door trip chains of people has a multitude of applications. For infrastructure planners and public transport providers knowledge of the true origins, destinations and volumes of commuters gives a much

better understanding of requirements for the road and transport networks than disconnected counts of cars or people at points of observation of the current network by using loops, cameras, ticket systems or manual passenger counting campaigns.

To assist users of public transport with relevant information about opportunities and problems it is vital to be able to generate a prediction of the next destination, time of travel and the means of public transport the person is going to use. For many people the same trips are repeated with regular cycles, i.e. daily, on (certain) weekdays, weekly or monthly. Such travellers can be proactively given targeted information about disruptions in road traffic or public transport lines, which they frequently use, at their personal times of regular usage. Real-time recognition of the current means of public transport and a prediction of the likely destinations can also be used to assist connecting passengers.

Multiple smartphone-assisted ways for automatic detection of public transport usage can be envisaged:

- *Ticketing system*: If coupled with the payment of the trip in systems where both vehicle entries and exits are registered, precise and correct information about the public transport part of trips can be obtained. Requires integration with the fare payment system.
- *Radio beacon*: Some transport providers may provide vehicle-installed radio transmitters, e.g. WiFi¹ or Bluetooth², which can be detected and compared with a list of beacon identifiers to detect the proximity of a public transport vehicle.
- *Live positioning*: The live positions of public transport vehicles can be matched with the measured positions of the passenger, searching for a sequence of continuous matches with suitable accuracy.
- *Static timetable*: Processed information about a trip carried out by a person (start and end locations and times, route geometry) using a vehicle can be compared with the information in a static timetable.

In our test arrangement we did not have access to the ticketing system of the public transport provider. Additionally, the current policy in the Helsinki region does not require registration of vehicle exits, and one validation of a regional ticket allows up to 80 min of transfer without a new validation. Therefore ticket-based information would not have been accurate even if it were available. At the time of testing listed Wi-Fi beacons were only available on trams, which does not give an adequate coverage of the transport network. Live positioning of public transport vehicles was made available by the public transport provider for a part of their fleet. Static timetable information was available for all lines and departures.

The target was to create a dataset for testing and benchmarking algorithms for automatic recognition of public transportation trips. The dataset is composed of position and activity recognition samples of 8 researchers between 9 am and

¹ <http://www.wi-fi.org/>.

² <https://www.bluetooth.com/>.

4 pm EET+DST on August 26th 2016, manual bookkeeping of their trips and the related transport infrastructure data. Data collection for the limited period was pre-agreed with every campaign participant to enable publication of the dataset without privacy concerns.

Seven participants executed as many public transportation trips as possible during the designated time, especially emphasising travel by subway, as it has been the most challenging transportation mode for automatic recognition. The eighth participant logged some private car trips to provide comparison data, which should not match with any public transportation.

Due to the exceptional amount of travel per person during one day this dataset cannot be used as a source for studying the regular travel habits of public transportation users. It also doesn't contain repeatedly visited locations such as homes or offices. The challenge is to correctly recognise as many trips listed in the manual log as possible by using the other forms of data available. The dataset consists of the following tables:

- *Device data*: samples from mobile device sensors.
- *Filtered device data*: selection of the perceived activity and exclusion of data points at times of staying still using our algorithms.
- *Device models*: phone models used by the participants.
- *Manual log*: manual trip bookkeeping entries of participants.
- *Live position samples*: public transport fleet positions.
- *Static timetables*: public transport timetables valid on the date of the experiment.
- *Train stop times*: measured train stop time information for the date of the experiment.

The complete dataset is available³ in *github*⁴.

2 Background

Transportation mode estimation and classification using mobile phone sensors has been discussed e.g. in [3, 5, 13]. Automatic recognition of bus trips using mobile phone sensors has earlier been addressed by the *Live+Gov*⁵ project. Their mobile client collected both hardware sensor (accelerometer, rotation vector, gyroscope, magnetic field) and processed activity detection data [6]. A project proprietary human activity recognition classifier was built and trained. Public transport recognition was attempted using public transport infrastructure data similar to our study (static timetables and live locations of public transport vehicles). The client software also supported user marking of activity, but users were shown the currently detected activity, and consequently they have generally registered their activities only in cases, where the detected activity was incorrect. Service line detection probability of 37% is reported [8].

³ Static timetable data is referenced from the resources of the transport provider.

⁴ <https://github.com/aalto-trafficsense/public-transport-dataset>.

⁵ http://cordis.europa.eu/project/rcn/102050_en.html.

In one study bus passengers with mobile phones have been used as voluntary crowdsourced sensors for reporting the live locations of buses [1]. In the Live+Gov project traffic jams were detected from irregularities in public transport fleet location data [6]. The specific context of parking has also been considered, estimating from mobile phone sensor data, whether a parking area has space available [9,11].

The present effort belongs to the *TrafficSense*⁶ project, a part of the *Aalto Energy Efficiency Research Programme*⁷. The project aims to save time and energy in traffic by understanding regular travel habits of individuals and proactively assisting them with information available in the internet. Sustainability and business model of the approach have been considered in [4].

The current TrafficSense mobile client is coded in native Android⁸. It is available on *Google Play*⁹, but currently restricted to Finland, because the public transport recognition and disruption information functions are only available locally. The main map view of the mobile application can show trips marked with the mode of transport, ranked regular destinations of the user, current traffic situation as well as the current location of the user. The mode of transport can be edited or confirmed by the application user. Public transport disruption bulletins matching the trip history and current location of the user are shown as notifications.

The types, formats and sources of open traffic information available were reviewed in [10]. For information on public transport the resources from *Helsinki Regional Transport*¹⁰ (HRT) were used. Their developer resources are currently provided and documented under the *digitransit*¹¹ umbrella. The resources follow the *SIRI* (Service Interface for Real-time Information [10]) specifications¹². Live locations of the fleet were sampled from their live data API¹³. The static timetable data is in *GTFS* (General Transit Feed Specification¹⁴) format as specified by Google Inc. Train stop time information encoded as *JSON* (Javascript Object Notation¹⁵ [2]) was fetched from the *digitraffic*¹⁶ service¹⁷ operated by the *Finnish Transport Agency*¹⁸ (FTA). All other sampled data is made available in *CSV* (Comma-Separated Values [12]) format. The repository includes

⁶ <http://trafficsense.aalto.fi>.

⁷ <http://aef.aalto.fi/en/>.

⁸ Android™ is a trademark of Google Inc.

⁹ <https://play.google.com/store/apps/details?id=fi.aalto.trafficsense.trafficsense>.

¹⁰ <https://www.hsl.fi/en>.

¹¹ <https://www.digitransit.fi/en/developers/>.

¹² <http://user47094.vs.easily.co.uk/siri/>.

¹³ At the time of the study, the data was available at <http://dev.hsl.fi/siriaccess/vm/json>. The address has been later replaced by <http://api.digitransit.fi/realtim/vehicle-positions/v1/siriaccess/vm/json>.

¹⁴ <https://developers.google.com/transit/gtfs/>.

¹⁵ <http://www.json.org/>.

¹⁶ <https://www.liikennevirasto.fi/web/en/open-data/digitraffic>.

¹⁷ http://rata.digitraffic.fi/api/v1/history?departure_date=2016-08-26.

¹⁸ <http://www.liikennevirasto.fi/web/en>.

scripts for importing the CSV tables into a *PostgreSQL*¹⁹ database, which is used internally by the TrafficSense project.

3 Manual Bookkeeping by Test Participants

All test participants manually documented the details of their trips during the day. The information provided for each trip leg is shown in Table 1. Because the timestamps were manually recorded, their values are approximate. In total 103 trips were recorded.

Table 1. Manually logged trip information (tz = timezone).

Label	Type	Description
<code>device_id</code>	integer	Device identifier, aligned with <code>device_id</code> in Sect. 4
<code>st_entrance</code>	string	Description of entrance to station building, if applicable A map of the letters to mark entrances at each subway station is provided in the repository
<code>st_entry_time</code>	timestamp (no tz)	Station entry time, if applicable
<code>line_type</code>	string	SUBWAY/BUS/TRAM/TRAIN/CAR
<code>line_name</code>	string	Identifier, e.g. 7A, 102T, U, V
<code>vehicle_dep_time</code>	timestamp (no tz)	Vehicle departure time
<code>vehicle_dep_stop</code>	string	Platform or other station where vehicle was boarded
<code>vehicle_arr_time</code>	timestamp (no tz)	Vehicle stop time at the end of the trip
<code>vehicle_arr_stop</code>	string	Platform or other station where the vehicle was exited
<code>st_exit_location</code>	string	Exit to station building, if applicable. Subway exit letters have been marked in maps provided in the repository
<code>st_exit_time</code>	timestamp (no tz)	Time of exiting station, if applicable
<code>comments</code>	string	Freeform comments about the trip leg

4 Mobile Device Measurements

Mobile device samples were collected using the TrafficSense mobile client. The client uses the fused location provider and activity recognition of *Google Play*

¹⁹ <https://www.postgresql.org/>.

*Services*²⁰. Both of them are *virtual sensors* [7], abstracting information from available hardware sensors²¹ into current best estimates of the location of the device and the activity the person carrying the device is currently engaged in. The sampled parameters contained in the dataset are listed in Table 2. The coordinates are in *WGS84*²² format. The table in the dataset contains 6,030 entries. It is formatted as CSV and sorted by `time` and `device_id`.

Table 2. Parameters sampled from mobile devices (tz = timezone).

Label	Type	Description
<code>time</code>	timestamp (no tz)	From the clock of the mobile device
<code>device_id</code>	integer	Stable identifier for the device
<code>lat</code>	double	Latitude, in WGS84
<code>lng</code>	double	Longitude, in WGS84
<code>accuracy</code>	double	Radius, in meters, estimated by the <i>fused location provider</i> of the mobile device
<code>activity_1</code>	enum	Activity with highest confidence, provided by <i>activity recognition of Google Play Services</i> . Values: IN_VEHICLE, ON_BICYCLE, RUNNING, STILL, TILTING, UNKNOWN, WALKING
<code>activity_1_conf</code>	integer	Percentage of recognition certainty, 100 = best confidence
<code>activity_2</code>	enum	Value of second-highest confidence activity
<code>activity_2_conf</code>	integer	Percentage of recognition certainty
<code>activity_3</code>	enum	Value of third-highest confidence activity
<code>activity_3_conf</code>	integer	Percentage of recognition certainty

4.1 Mobile Client Filtering Algorithms

The client alternates between **ACTIVE** and **SLEEP** states. The current default value for the sleep timer is 40 s. If the detected position changes by a distance longer than the accuracy of the position fix during a period of the perceived activity indicating **STILL**, the timer is restarted. This rule aims to prevent erroneous transitions to **SLEEP** state if the device is moving, even if activity detection perceives it as being still. Such situations typically occur during smooth trips on rails; i.e. trains, trams and subways.

In **ACTIVE** state position is requested from the *fused location provider*²³ provided by Google Play Services with *high accuracy* and a 10 s interval, which yields

²⁰ <https://developers.google.com/android/guides/overview>.

²¹ The availability of a particular hardware sensor may vary between different device models.

²² <http://gisgeography.com/wgs84-world-geodetic-system/>.

²³ <https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi>.

a sufficiently accurate description of the route up to motorway speeds (333 m sample interval at 120 km/h). In **SLEEP** state position requests are dropped to *no power* priority, which means that position fixes are passed to our client only if requested by another application. Activity reports are always requested with a 10 s interval, but as a form of device power saving, during **STILL** detections activity recognition interval has been observed to increase up to 180 s. As a result, sometimes the client may need up to ≈ 200 s of movement to wake up from **SLEEP** state.

In our data format each accepted position record is coupled with the latest activity information. The timestamp of the entry is the timestamp of the position, not the activity detection. Therefore the same detected activity may repeat over multiple points. The received position fixes ('points') are filtered as follows (Fig. 1):

- If 60 min have passed since the last accepted point, any point is accepted (to provide a "ping" effect and record that the client was running).
- Accuracy²⁴ must be better than 1000 m.
- If `activity != last queued activity` and `activity` is 'good'²⁵, the point is accepted.
- If `(activity == last queued activity)` and `(distance to last accepted point > accuracy)`, the point is accepted.

Unless stated otherwise above, the parameters have been chosen experimentally, searching for balance between accurate trip data, low mobile phone battery consumption and reasonable quantity of samples. Adapting the location sample rate to the velocity of the mobile device could further improve efficiency especially if satellite positioning can be switched off between samples. The biggest challenge

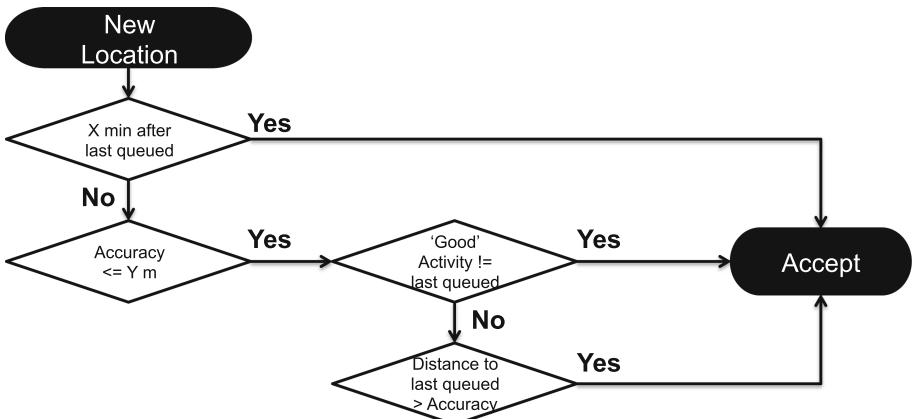


Fig. 1. Filtering algorithm for incoming position fixes.

²⁴ Estimated by the *fused location provider*.

²⁵ Not UNKNOWN or TILTING.

was to keep the client from going to **SLEEP** during activity. The issue could be mitigated by a longer **SLEEP** timer at the cost of increased power consumption. A better approach would be to improve the accuracy of the activity recognition. Using the chosen parameters, the resulting dataset contains about 30% of the theoretical maximum²⁶ number of points.

The fused location provider used by the mobile client combines data from satellite, Wi-Fi and cellular positioning. Despite that, sometimes positioning errors occur. In a typical case the correct position alternates with a distant point, probably due to changing between different positioning methods within the fused location provider.

4.2 Filtered Device Data

The dataset also includes a table of filtered `device_data`. The activity recognition in the received trace is noisy with often occurring spurious transitions, which makes it impractical to use those segments for matching to public transport directly. The second level filtering operation generating `device_data_filtered` serves to refine more stable contiguous activity segments from the original trace.

The received activity confidences are summed over a sliding two minute window around each point to identify a dominant activity. Furthermore six consecutive samples—typically one minute at the ten second sampling interval—of the same activity are required to change states, splitting any indeterminate region between consecutive stable activities. In addition, points more than five minutes apart in the trace, either due to the terminal being stationary or lacking location reception, causes a segment break. Segments where no stable activity is found, are omitted. These steps eliminate activity segments of less than a minute, that are unlikely to represent, or be accurately matchable to, public transport.

Filtered device data is included in the published data set, because some of our recognition algorithms use it. A new candidate solution is welcome to base itself on the more complete `device_data` instead, and implement other filtering approaches. The activity filtering algorithm has a clear impact on recognition results, as can be seen from the data shown in Sect. 5.1. The following parameters are included in `device_data_filtered` (corresponding descriptions are the same as in Table 2):

1. `time` (timestamp no tz)
2. `device_id` (integer)
3. `lat` (double, latitude)
4. `lng` (double, longitude)
5. `activity` (enum value of the winning activity)

The table contains 5,975 points. The CSV-version is sorted by `time` and `device_id`.

²⁶ One point every 10 s from every terminal.

4.3 List of Device Models

A list of the models of the eight smartphones used by the test participants is provided as a separate table. It is included, because some differences were observed in e.g. activity recognition performance between different devices. The table contains the following columns:

1. `device_id` (integer, same as in Table 2)
2. `model` (string name of the model)

5 Public Transport Infrastructure Information

Details of the information sourced from public transport infrastructure providers is described in this section.

5.1 Live Positions of Public Transport Vehicles

The live positions of the public transport fleet were obtained from HRT and sampled at 30 s intervals. The columns recorded into the dataset are shown in Table 3. The time period was restricted to the time of the trial. The maximum and minimum coordinates recorded by the participants were checked and the live vehicle position data was filtered (as a rectangle) to include only the area surrounding the locations sampled from the test participants. The table length is 229,451 entries.

Table 3. Live positions of public transport vehicles (`tz` = timezone). All data as provided by the transit live data API.

Label	Type	Description
<code>time</code>	timestamp (no <code>tz</code>)	The time the position was recorded
<code>lat</code>	double	Vehicle location latitude, in WGS84
<code>lng</code>	double	Vehicle location longitude, in WGS84
<code>line_type</code>	enum	One of: <code>SUBWAY/BUS/TRAM/TRAIN/FERRY</code>
<code>line_name</code>	string	Identifier for the public transport line, e.g. 7A, 102T
<code>vehicle_ref</code>	string	Distinguish between different vehicles with the same <code>line_name</code> in traffic at the same time

5.2 Static Timetables

The static timetables from HRT are not included in the data repository, but the file is available through the following link: http://dev.hsl.fi/gtfs/hsl_20160825T125101Z.zip. It can be used e.g. with an *OpenTripPlanner*²⁷ (OTP) server to query for trips with matching start and end locations and start times. A more precise description for the specific task of querying timetable information can be found in the documentation for the digitransit²⁸ API. The data includes the static timetables for all the public transport vehicles (also local trains) used by the study participants.

5.3 Train Stop Times

The dataset also includes information about the recorded stop times of local trains at stations on the day of the study. The information²⁹ includes a JSON-format `junat` object, including the stopping times of trains at each station as described (in Finnish) at <http://rata.digitraffic.fi/api/v1/doc/index.html#Junavastaus>. Information on the referenced stations, including their locations, can be obtained as <http://rata.digitraffic.fi/api/v1/metadata/stations>. The description of the stations “Liikennepaikat” format is available (in Finnish) at <http://rata.digitraffic.fi/api/v1/doc/index.html#Liikennepaikkavastaus>.

Table 4. Manually logged bus trips, which are **not** available in live transit data.

device_id	Departure time	line_name
1	09:07:00	154
5	09:08:00	110
5	09:16:00	18 (Espoo)
3	09:59:32	132T
2	10:38:51	103T
6	13:26:00	95
2	13:34:35	103T
3	14:21:08	105
3	15:26:39	102T
1	15:59:00	156

²⁷ <http://www.opentripplanner.org/>.

²⁸ <https://www.digitransit.fi/en/developers/services-and-apis/1-routing-api/itinerary-planning/>.

²⁹ <https://github.com/aalto-trafficense/public-transport-dataset/tree/master/trains.json>.

6 Data Coverage and Limitations

No local trains and not all the buses are included in the live transit data. Travelled `line_name` specifiers (appearing in the manually logged data) **found** in `transit_live` are:

- Trams: 2, 3, 7A, 8, 9
- Buses: 16, 67, 72, 550, 560
- Subways (line_name in manual-log varies and cannot be used for comparison)

Table 5. Manually logged trips correctly recognised from live data (28) using our algorithms (logged trips matching multiple segments have multiple rows).

dev_id	log_start	log_end	log_type	log_name	id	segm_start	segm_end	activity	recd_type	recd_name
1	09:41:00	09:44:00	TRAM	3	6	09:40:58	09:44:59	IN_VEHICLE	TRAM	3
1	10:06:00	10:19:00	TRAM	7A	12	10:07:24	10:13:23	IN_VEHICLE	TRAM	7A
1	10:06:00	10:19:00	TRAM	7A	13	10:13:34	10:14:56	ON_BICYCLE		
1	10:06:00	10:19:00	TRAM	7A	14	10:15:37	10:20:11	IN_VEHICLE	TRAM	7A
1	10:46:00	10:52:00	SUBWAY	V	19	10:43:41	10:52:45	IN_VEHICLE	SUBWAY	V
1	11:01:00	11:06:00	SUBWAY		21	11:01:12	11:09:28	IN_VEHICLE	SUBWAY	V
1	11:15:00	11:19:00	SUBWAY	M	22	11:16:07	11:20:23	IN_VEHICLE	SUBWAY	M
1	13:14:00	13:30:00	SUBWAY		29	13:13:31	13:30:27	IN_VEHICLE	SUBWAY	V
1	14:18:00	14:30:00	SUBWAY		34	14:17:32	14:22:30	IN_VEHICLE	SUBWAY	V
1	14:18:00	14:30:00	SUBWAY		35	14:22:41	14:26:37	ON_BICYCLE		
1	14:18:00	14:30:00	SUBWAY		36	14:27:18	14:31:15	IN_VEHICLE	SUBWAY	V
1	14:38:00	14:51:00	SUBWAY		37	14:31:25	14:39:23	WALKING		
1	14:38:00	14:51:00	SUBWAY		38	14:39:54	14:51:15	IN_VEHICLE	SUBWAY	V
1	14:57:00	15:05:00	SUBWAY		40	14:56:47	15:06:45	IN_VEHICLE	SUBWAY	V
1	15:36:00	15:55:00	SUBWAY		44	15:36:27	15:42:17	IN_VEHICLE	SUBWAY	M
1	15:36:00	15:55:00	SUBWAY		45	15:42:27	15:48:05	ON_BICYCLE		
1	15:36:00	15:55:00	SUBWAY		46	15:50:21	16:18:46	IN_VEHICLE		
2	13:14:34	13:30:03	SUBWAY	To west	54	13:14:01	13:31:34	IN_VEHICLE	SUBWAY	V
3	10:59:27	11:10:08	SUBWAY	To east	74	10:59:43	11:09:46	IN_VEHICLE	SUBWAY	V
3	11:27:06	11:34:11	SUBWAY	To west	76	11:28:05	11:34:36	IN_VEHICLE	SUBWAY	V
3	13:14:14	13:24:28	SUBWAY	To west	81	13:15:14	13:23:37	IN_VEHICLE	SUBWAY	V
4	10:17:00	10:28:00	TRAM	7A	105	10:14:27	10:38:15	IN_VEHICLE	TRAM	7A
4	10:47:00	10:53:00	SUBWAY	to east	107	10:45:59	10:53:39	IN_VEHICLE	SUBWAY	V
4	13:21:50	13:31:00	SUBWAY	to west	116	13:21:20	13:31:57	IN_VEHICLE	SUBWAY	V
4	13:44:00	13:47:00	SUBWAY	to west	118	13:45:12	13:47:30	IN_VEHICLE	SUBWAY	V
4	14:28:00	14:31:00	TRAM	9	122	14:26:04	14:31:10	IN_VEHICLE	TRAM	9
5	10:52:00	11:02:00	BUS	16	136	10:52:30	11:03:28	IN_VEHICLE	BUS	16
5	14:40:00	14:50:00	SUBWAY	R	149	14:40:46	14:50:51	IN_VEHICLE	SUBWAY	V
5	15:10:00	15:20:00	TRAM	9	151	15:08:38	15:24:01	IN_VEHICLE	TRAM	9
6	14:01:00	14:17:00	BUS	560	162	13:59:21	14:25:50	IN_VEHICLE	BUS	560
8	10:18:00	10:23:00	TRAM	9	174	10:17:00	10:23:55	IN_VEHICLE	TRAM	9
8	10:44:00	10:53:00	TRAM	9	178	10:44:22	10:54:44	IN_VEHICLE	TRAM	9
8	11:08:00	11:13:00	BUS	72	180	11:07:46	11:14:48	IN_VEHICLE	BUS	72
8	11:25:00	11:41:00	SUBWAY		181	11:26:23	11:42:24	IN_VEHICLE	SUBWAY	M
8	13:49:00	14:04:00	BUS	550	187	13:48:41	14:04:30	IN_VEHICLE	BUS	550

Travelled `line_name` specifiers **not found** in `transit-live` are:

- Trains: E, I, K, P, U
- Buses: Espoo18³⁰, 95, 102T, 103T, 105, 110, 132T, 154, 156

Table 6. Manually logged trips having a₁ corresponding IN_VEHICLE segment in sampled data (38), but not correctly recognised from `transit_live` by the current algorithms.

dev_id	log_start	log_end	log_type	log_name	id	segm_start	segm_end	activity	recd_type	recd_name
1	09:31:00	09:35:00	SUBWAY	V	4	09:31:44	09:36:37	IN_VEHICLE		
1	09:48:00	09:51:00	SUBWAY		8	09:49:19	09:54:48	IN_VEHICLE		
1	10:28:00	10:35:00	SUBWAY		17	10:28:00	10:37:33	IN_VEHICLE		
1	11:42:00	11:43:00	SUBWAY	M	24	11:40:36	11:46:04	IN_VEHICLE		
1	11:56:00	12:01:00	SUBWAY		26	11:55:14	12:02:53	IN_VEHICLE		
1	13:43:00	13:52:00	TRAM	9	31	13:41:40	13:46:33	ON_BICYCLE		
1	13:43:00	13:52:00	TRAM	9	32	13:46:54	14:05:12	IN_VEHICLE		
1	13:58:00	14:04:00	SUBWAY		32	13:46:54	14:05:12	IN_VEHICLE		
1	15:24:00	15:26:00	SUBWAY		42	15:24:38	15:28:24	IN_VEHICLE		
2	11:04:00	11:06:33	SUBWAY	To east	50	11:04:13	11:12:20	IN_VEHICLE		
2	11:26:18	11:50:00	SUBWAY	To east	51	11:22:41	11:58:31	IN_VEHICLE		
2	11:50:44	11:58:40	SUBWAY	To west	51	11:22:41	11:58:31	IN_VEHICLE		
3	10:21:21	10:23:01	SUBWAY	To east	70	10:17:28	10:25:54	IN_VEHICLE		
3	10:37:47	10:47:59	SUBWAY	To east	72	10:36:03	10:48:14	IN_VEHICLE		
3	11:56:11	11:58:43	SUBWAY	To west	78	11:57:56	12:01:18	IN_VEHICLE		
3	13:35:13	13:41:37	TRAM	7A	83	13:28:48	13:41:28	IN_VEHICLE		
3	13:45:37	13:47:12	SUBWAY	To west	85	13:45:42	13:48:24	IN_VEHICLE		
3	14:07:59	14:09:23	SUBWAY	To west	87	13:59:48	14:14:14	IN_VEHICLE		
3	14:11:30	14:13:34	SUBWAY	To west	87	13:59:48	14:14:14	IN_VEHICLE		
3	14:42:40	14:52:31	SUBWAY	To east	90	14:39:47	14:52:22	IN_VEHICLE		
3	15:02:03	15:13:57	SUBWAY	To west	92	15:01:12	15:16:46	IN_VEHICLE		
4	09:39:00	09:45:00	TRAM	9	97	09:38:15	09:41:02	IN_VEHICLE	TRAM	3
4	09:39:00	09:45:00	TRAM	9	98	09:41:12	09:42:25	ON_BICYCLE		
4	09:39:00	09:45:00	TRAM	9	99	09:43:08	09:45:15	IN_VEHICLE	TRAM	1
4	10:35:00	10:37:00	SUBWAY	to east	105	10:14:27	10:38:15	IN_VEHICLE	TRAM	7A
4	11:04:00	11:13:00	SUBWAY	to east	109	11:04:55	11:14:10	IN_VEHICLE		
4	11:26:00	11:28:00	SUBWAY	to west	111	11:23:59	11:31:36	IN_VEHICLE		
4	11:36:00	11:38:00	SUBWAY	to east	113	11:36:32	11:46:49	IN_VEHICLE		
4	11:45:00	11:47:00	SUBWAY	to east	113	11:36:32	11:46:49	IN_VEHICLE		
4	13:57:00	14:01:00	SUBWAY	to west	120	13:57:00	14:02:39	IN_VEHICLE		
4	15:15:00	15:20:00	SUBWAY	to east	125	15:15:29	15:20:13	IN_VEHICLE		
5	09:58:00	10:08:00	TRAM	7A	133	09:58:03	10:14:13	IN_VEHICLE		
5	10:12:00	10:13:00	SUBWAY	M	133	09:58:03	10:14:13	IN_VEHICLE		
5	11:11:00	11:14:00	SUBWAY	R	138	11:09:19	11:14:28	IN_VEHICLE		
5	11:18:00	11:20:00	BUS	67	140	11:18:23	11:20:53	IN_VEHICLE		
5	11:38:00	11:52:00	SUBWAY	M	142	11:38:53	11:50:59	IN_VEHICLE		
5	15:36:00	15:39:00	TRAM	9	153	15:37:43	15:40:10	IN_VEHICLE		
6	11:38:00	11:52:00	SUBWAY		156	11:45:26	11:50:26	IN_VEHICLE		
6	14:19:00	14:25:00	SUBWAY		162	13:59:21	14:25:50	IN_VEHICLE	BUS	560
6	15:50:00	15:59:00	TRAM	9	165	15:50:58	15:59:24	IN_VEHICLE		
8	11:52:00	11:58:00	SUBWAY		183	11:52:35	12:00:18	IN_VEHICLE		

³⁰ Bus `line_name` specifiers <100 can be re-used by the adjoining cities of Helsinki, Espoo and Vantaa. Bus 18 from Helsinki is included in `transit_live`, but the test participant used bus 18 of Espoo.

In terms of recorded trips, the 10 bus trips shown in Table 4 can therefore definitely not be found in `transit_live`. The 28 trips in Table 5 can be confirmed as being discoverable and identifiable using transit live data, as they have been found with our algorithm.

The 38 trips in Table 6 have detected `IN_VEHICLE` (sometimes also `ON_BICYCLE`) segments in `device_data_filtered` overlapping with entries in `manual_log`. Therefore these trips should be recognisable if the particular vehicle exists in `transit_live`, but as shown in the `recd_type` and `recd_line` columns, they have not been correctly recognised by the current algorithms. Logged trips matching multiple recorded segments (e.g. 09:39 trip with tram 9 matching segments 97, 98 and 99) are listed separately for each segment. For performance gains in recognising trips using live data, these are expected to be the best candidate trips.

Finally, the 13 trips in Table 7 have no overlapping `IN_VEHICLE` segment from the mobile device samples. It is also noted that:

- 3/13 trips (2 subway + 1 tram) have no overlapping mobile device samples at all
- 10/13 trips overlap with `WALKING` (in 6/10 cases shadowing the whole trip)
- 12/13 trips were carried out with the same device model (`device_id` 5 and 6)

The `device_data` sampled before and after a time segment with no data should still reveal the mobile device displacement without recording positions, i.e. it has either erroneously been in `SLEEP` state or no position fixes have been received. Looking at the timestamps and locations recorded before and after the break period with geographical displacement, a set of candidate options could still be obtained from either the static timetable or live fleet position data. In downtown

Table 7. Manually logged trips having no corresponding `IN_VEHICLE` segment (13) in sampled data.

dev_id	log_start	log_end	log_type	log_name	id	segm_start	segm_end	activity
4	15:53:00	15:55:00	SUBWAY	to west	127	15:41:39	15:58:40	WALKING
5	11:30:00	11:31:00	SUBWAY	R	141	11:21:26	11:36:31	WALKING
5	13:22:00	13:28:00	SUBWAY	M	144	13:17:08	13:23:38	WALKING
5	13:34:00	13:43:00	SUBWAY	R	145	13:28:54	13:37:27	WALKING
5	13:54:00	14:02:00	SUBWAY	V				
5	14:20:00	14:30:00	SUBWAY	R	148	14:02:42	14:39:19	WALKING
5	15:36:00	15:39:00	TRAM	9	152	15:34:32	15:37:11	WALKING
5	15:51:00	15:52:00	TRAM	8				
5	15:56:00	15:57:00	SUBWAY	M				
6	13:47:00	13:49:00	SUBWAY		161	13:44:23	13:57:30	WALKING
6	14:29:00	14:30:00	SUBWAY		163	14:26:10	15:03:54	WALKING
6	15:02:00	15:20:00	SUBWAY		163	14:26:10	15:03:54	WALKING
6	15:41:00	15:43:00	TRAM	2	164	15:27:41	15:50:48	WALKING

areas with many transport alternatives errors would be likely, but in less busy areas the set of alternatives will be smaller. Faulty recognitions changing car trips to public transport would still be very likely, so this type of recognition should only be attempted in scenarios where the penalty of false positive is not very high. This type of recognition has not been attempted in the current experiment.

7 Methods of Automatic Recognition

The collected live public transport vehicle locations and static timetables, in conjunction with the user location traces, were used to automatically recognise public transport trips taken by users. Trip legs consisting, after filtering, of a continuous sequence of IN_VEHICLE activity were matched against the vehicle position traces and timetable data. The train stop times were not used.

7.1 Live Vehicle Location

The vehicle location data from Table 3 is compared with the user locations collected by the personal mobile devices as described in Sect. 4. Potential issues in matching include:

- Missing vehicle or user data points
- Inaccurate location points
- Clock differences
- Distance between user and vehicle location sensor in longer vehicles
- Distance between location samples at higher vehicle velocities
- False matches to other public transport vehicles
- False positives where a car trip takes place near public transport vehicles
- Intermittent changes of the line name label on some vehicles in the live data

For performance reasons the number of user location samples used for matching a trip leg was limited to 40. To counteract the location accuracy and vehicle length issues, a distance limit of 100 m was used for collecting vehicle matches. A greater limit may cause more false positive matches to appear.

With the sample rate of thirty seconds in the collected vehicle positions, a vehicle traveling at 80 km/h would produce samples every 667 m, much in excess of the above mentioned distance limit. This could cause false negatives with the user location samples falling in between the vehicle points in such a way that they are not matched. To prevent this, the position sequence of each vehicle is processed into linestrings, and user point distances calculated against those line geometries.

The vehicle location points for composing the linestrings for comparison are collected in a $\pm 60\text{-s}$ window around the timestamp of each user point sample. This allows for some clock difference, and sampling time difference.

For a vehicle to be accepted as a possible match, its linestring must be within the 100 m distance limit for a minimum of 75% of the user location samples.

Each match within the permitted 100 m distance accumulates a score of $100 - d$ when the distance is d metres. The vehicle with the highest score wins.

The line type and name are set according to the matched vehicle. In case of the vehicle having multiple line names in the matches, the most frequently occurring name is used. On some lines, a vehicle can intentionally change line name when passing a certain stop. Also, some vehicles in the data set change line name intermittently to false values.

In addition to the method described above (subsequently referred to as “New live”), a prior implementation (“Old live”) was evaluated. In the older implementation, four user trace sample points were used, and matched against vehicle location points in the surrounding time window. With the vehicle location sampling interval of 30 s, and the 100 m distance criterion, this would be expected to cause otherwise optimal user trace point samples to potentially fall outside the matching radius once the vehicle speed exceeds $2 \times 100 \text{ m}/30 \text{ s} \approx 6.67 \text{ m/s} \approx 24 \text{ km/h}$.

For trams in Helsinki having an average speed of 14.7 km/h (2013–2014)³¹, with dense stops the new and old methods should produce similar results. More differences can be expected on the subway, and on bus routes with highway segments, where speeds are higher and stops more sparse.

7.2 Matching with Static Timetables

Comparison with static timetables is based on searching for public transport plans of past trips using the sampled IN_VEHICLE segments. Firstly, a trip-plan query is sent to the OTP journey planner interface based on each candidate leg’s start-time, first point (origin) and last point (destination). Secondly, the resulting PT plans are compared to the leg’s start-time and end-time as well as the user location trace to identify whether the leg represents a PT ride. Potential issues in matching include:

- Missing or inaccurate user location points
- Inaccuracy in activity determination and filtered transition points
- Clock differences
- False positives when the route and timing of a car trip are similar to a segment of a public transport line

Table 8 explains the constants and Table 9 the variables and formulas used in the process. Figure 2 illustrates an example public transport trip and highlights the following parameters showing possible extra parts of the planned trip compared to the original sampled trip: tWb , tWe , $tPTb$, $tPTe$, where t indicates “time”, W indicates “walking” and PT indicates “public transport”. The b denotes “at the beginning of a trip or leg”, and e denotes “at the end of a trip or leg”.

The data sampled from a mobile device can often have a fair amount of inaccuracy in location and activity detection. For this reason the filtered transition points starting and ending the vehicular trip leg can often appear between stops during the actual mass transit leg. Therefore almost all travel plans from the

³¹ https://www.hsl.fi/sites/default/files/uploads/13_2015_raitoliikenteen_linjastosuunnitelma_netti.pdf.

Table 8. Constants used in our query to the Open Trip Planner, validation and matching of the resulting trip plan with the recorded trip.

Parameter	Definition	Value
$dEmax$	Maximum acceptable distance inaccuracy as a result of displacement or delay in detecting the point of mode transition. This value denotes an offset in start point and/or end point of filtered IN_VEHICLE leg	500 m
vW	Walking speed	$\leq 1.34 \text{ m/s}$
vPT	Speed of a PT vehicle	$\geq 3 \text{ m/s}$
$tEPT$	Deviation of the PT vehicle from schedule	$\leq 3 \text{ min}$

Table 9. Variables and formulas used in our query to the Open Trip Planner, validation and matching of the resulting trip plan with recorded trip.

Par.	Definition	Formula	Value
tV	Duration of the sampled IN_VEHICLE trip leg		
t	Duration of the planned trip		From OTP
dEb	Distance offset (error) in detecting the starting point of a leg		$\leq dEmax$
dEe	Distance offset (error) in detecting the end point of a leg		$\leq dEmax$
tWb	Walking duration at the beginning of the planned trip to reach the boarding stop	dEb/vW . For maximum acceptable inaccuracy: $tWb = 500/1.34 = 6.2 \text{ min}$	$\leq 6.2 \text{ min}$
tWe	Walking duration at the end of the planned trip from the end stop to the destination	dEe/vW	$\leq 6.2 \text{ min}$
tPT	Duration of the PT leg included in the planned trip		From OTP
$tPTb$	Extra transit ride at the beginning of the planned trip compared to the sampled IN_VEHICLE trip leg	dEb/vPT	$\leq 2.8 \text{ min}$
$tPTe$	Extra transit ride at the end of the planned trip compared to the sampled IN_VEHICLE trip leg	dEe/vPT	$\leq 2.8 \text{ min}$
ΔtPT	Extra transit ride of the planned trip compared to the sampled trip	$ tPT - tV = tPTb + tPTe$	$\leq 5.6 \text{ min}$
ΔtW	Extra walking in the planned trip compared to the sampled trip	$tWb + tWe$	$\leq 12.4 \text{ min}$
Δt	Duration difference between the planned trip and the sampled IN_VEHICLE leg	$\Delta tPT + \Delta tW = tWb + tPTb + tPTe + tWe$	$\leq 18 \text{ min}$

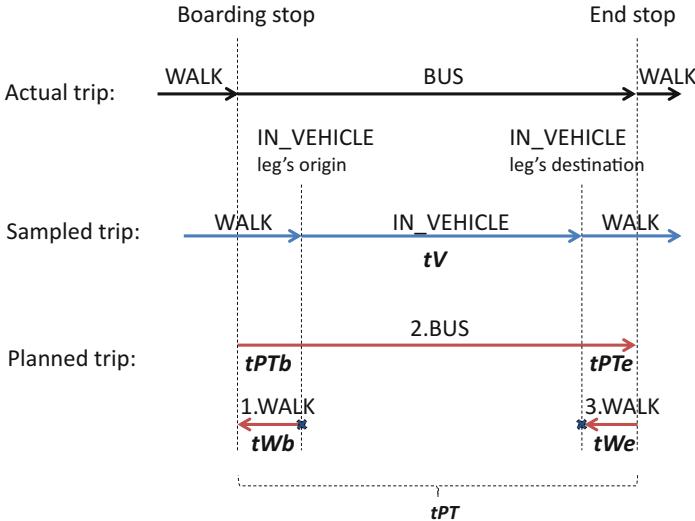


Fig. 2. Matching a sampled trip with the static public transport timetable to detect whether or not the trip has been a PT trip as well as to identify the PT mode (e.g. Tram, Bus, Commuter Train).

journey planner inevitably include walking sections in addition to the desired transit (vehicular) leg. The additional walking sections can appear at the beginning (tWb to walk from the detected start of vehicular leg to the public transport stop for boarding) and/or at the end of the trip leg (tWe to walk from the exit stop to the detected end of vehicular leg).

Selection of Query Parameters. The origin and destination parameters of our OTP query are set equal to the start and end geolocations of the sampled leg, respectively. The third query parameter, trip start-time, is set based on the leg's start-time as follows: The maximum allowable inaccuracy in transition points is $dEmax = 500$ m. Therefore, to account for inaccuracy in time and place of origin, in our query to the journey planner the earliest permissible start-time of the trip is adjusted back by $tWb = 6.2$ min to allow 500 m of walking at a speed of $vW = 1.34$ m/s to the boarding stop. Figuratively speaking, in case of offset in recorded IN_VEHICLE leg, we allow the traveller to catch the desired transit line according to actual timetable by walking to the nearest stop. If there is no delay and time and place of transition are completely accurate, the traveller would just wait 6.2 min at the boarding stop to take the transit line. Adjacent stops are assumed to be no more than one kilometre (2×500 m) apart.

Correspondingly, to relax both the origin and the destination, a maximum of $2 \times dEmax = 1,000$ m of walking is requested for each resulting plan. The OTP query requests for three PT plans for each candidate vehicular leg.

Filtering and Validation of Query Response. Out of the three returned plans the best match, if good enough, is chosen. Plans that match the following criteria can be *discarded*:

- Plans having a total duration of more than $tEPT = 3$ min shorter than the user-recorded vehicle leg, thereby assuming that the transit vehicle must travel closely according to schedule to avoid false positive matches.
 - Plans having a total duration of more than $\Delta dt = 18$ min longer than the user-recorded vehicle leg. This difference denoted by Δdt includes the walk times for dealing with location inaccuracy discussed above ($\Delta tW = 6.2$ min $\times 2 = 12.4$ min for the whole trip), and the time assumed for a mass transit vehicle to travel between adjacent stops (maximum 1000 m for the whole trip at minimum speed of $vPT = 3$ m/s, equal to $\Delta tPT = 5.6$ min).
 - Plans where the duration of the included transit leg (tPT) differs by more than $\Delta tPT = 5.6$ min from the user-recorded vehicular leg (tV), based on assumed time of travel of the vehicle between adjacent sparse stops.
 - Plans where the start of the included transit leg differs from that of the recorded vehicular leg by more than 5.8 min. This value considers a public transport vehicle traveling between adjacent stops at the beginning of the trip ($tPTb = 2.8$ min, for 500 m) plus an assumed $tEPT = 3$ min deviation of the transit line from its schedule.

The plans returned by the journey planner interface include location point sequences of the planned trips. As illustrated in Fig. 3, the location point sequences are also matched against the recorded user trace to verify that the

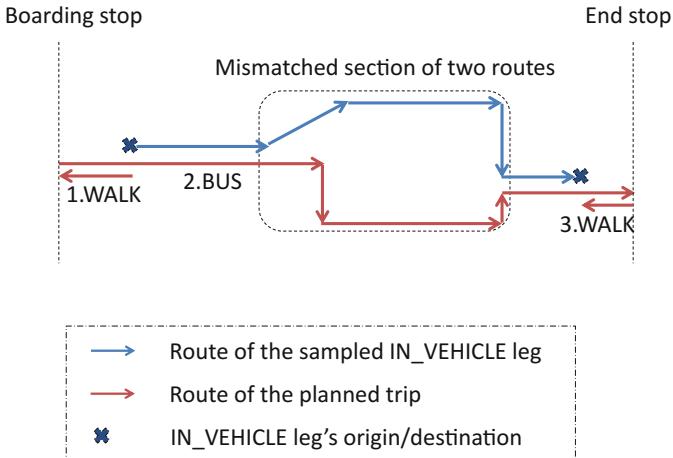


Fig. 3. Matching the sampled trip route with the planned PT route. This figure illustrates an example of a mismatch between routes of planned PT transit and the sampled vehicular leg. PT vehicle of the planned trip takes a different route by partly passing different roads compared to the original sampled trip.

route of the public transport line matches the recorded user trace. For comparison the recorded user trace is sampled at no less than 100 m intervals, and leading and trailing points may be ignored based on the assumed inaccuracy of origin and destination. A minimum of 70% of sample points must match a plan point within 100 m, and no more than four adjacent sample points may fall outside 100 m of the plan points for the plan to qualify. Out of the qualifying plans, the one with the closest start-time to that of the recorded leg wins.

Our current algorithm only validates plans containing a single vehicular leg. However, quick transfers between vehicles may not have been detected as separate vehicular legs by the activity detection and filtering of the mobile device. Allowing trip plans with transfers, and splitting the reference vehicular leg accordingly, could produce improved detection for such cases. An example of the fusion of a **BUS** leg and a **SUBWAY** leg to a single **IN_VEHICLE** leg can be found from device 6 at 13:59 to 14:25, where a short transfer has occurred between 14:17 and 14:19.

8 Recognition Results

Compared with the 103 manually logged trips, 86 **IN_VEHICLE** segments were recognised by our filtering algorithm, 85 of them overlapping at least a part of a logged trip.

The 28 trips recognised with correct `line_name` from `transit_live` using the new live algorithm were detailed in Table 5. The statistics for all the recognition methods are collected to Table 10. The static approach yielded the highest number of matched trips, but with one false `line_type` and two false bus `line_names`. Live recognition performance suffered from the absence of many buses and all trains in the data, but especially subway trip detection was significantly improved

Table 10. Trip matching statistics for all recognition methods compared to the manually logged trips (“line name” = matching line name, “line type” = matching line type).

	Static	Old live	New live	Combined	Logged
Bus	8	3	4	9	15
Bus (line name)	6	3	4	7	15
Tram	8	8	8	9	15
Tram (line name)	8	7	7	9	15
Train	4	0	0	4	9
Train (line name)	4	0	0	4	9
Subway	19	9	17	25	58
Public transport	40	20	29	48	97
Public transport (line type)	39	20	29	47	97

with the new live algorithm compared with the old one. Looking at the combined results but without requiring the `line_name` to match, 60% of bus and tram trips, 44% of train trips and 43% of subway trips were recognised. If the correct `line_name` is required, bus recognition success drops to 47%.

9 Discussion and Conclusions

The referenced dataset describes the trips of seven study participants using public transportation in the Helsinki area for a day, and a reference participant using a private car. The dataset includes a manual log of the trips, automatically collected measurements from the mobile devices of the participants and the public transport infrastructure data, which was available from the public transport provider at the time of the test. The mobile device measurements include geographical position and activity estimates. The infrastructure data consists of live locations of public transport vehicles and static timetable data. The challenge is to correctly match as many public transport trips as possible using the various measurements. The results can be verified using the manual log.

The data suffers from multiple imperfections. The estimated activities from the mobile devices are not always correct, e.g. sometimes a passenger is perceived to be riding a bicycle during a tram trip. Due to the power saving features of the mobile device client application and problems in positioning in trains and underground scenarios, measurements can be intermittent or completely missing. Other problems in positioning sometimes result in locations hopping between the correct position and a single distant point. Live locations were not available from all public transport vehicles at the time of the trial.

Altogether 103 trips are logged in the dataset, 97 of them carried out using public transportation. Combining matches from the static timetable and live data 60% of bus and tram trips, and 43–44% of train and subway trips were recognised with the correct vehicle type. Recognition of correct line names was otherwise on the same level, but for buses the recognition result dropped to 47%. The joint combined public transport recognition reached a level of 48% for the correct line type.

The currently achieved results are approximate, but adequate for purposes, where exact recognition of every trip is not necessary. For e.g. public transport disruption information filtering the current recognition level would be sufficient, because the likelihood of a correct recognition increases fast for frequent trips on the same public transport line and the penalty for false positive recognitions is not very high. For purposes where the passenger can review past trips individually the current level of performance can be frustrating. For fare payment processing it would be unacceptable.

Improvements in activity recognition accuracy of the mobile device and better vehicle coverage of live transit data would both contribute to better recognition probability. With the current power saving algorithms the mobile device power consumption is reasonable but clearly noticeable, especially when the device is constantly in motion, causing positioning to be requested with high accuracy.

Without radical improvements in battery and/or positioning technologies power consumption should be decreased rather than increased, which sets limitations to future improvements in the client sampling. One approach for testing would be to make more use of radio beacons in public transport vehicles. While in proximity of an identified radio beacon, the high accuracy positioning of the mobile device could be switched off and the system would rely on the positioning service of the public transport vehicle.

Acknowledgements. Supported by the TrafficSense project in the Aalto Energy Efficiency Programme funded by Aalto University.

In addition to authors the following persons have contributed to the TrafficSense software: Joonas Javanainen, Kimmo Karhu, Juho Saarela, Janne Suomalainen, Michailis Tziotis. In addition to the authors and software contributors the TrafficSense project has been participated by Mikko Heiskala, Jani-Pekka Jokinen, Iisakki Kosonen, Esko Nuutila, Roelant Stegmann, Markku Tinnilä, Seppo Törmä, Görkem Yetik and Indre Zliobaite.

Licensing

The data in the referenced github repository is licensed under the *Creative Commons BY 4.0 licence*³². The data extracted from the test group³³ is licensed by Aalto University. Train data³⁴ is obtained from Digitraffic offered by the Finnish Transport Agency. The live data on public transportation³⁵ is compiled from a service offered by Helsinki Regional Transport. Map data from the city of Helsinki³⁶ is authored by “Helsinki, kiinteistöviraston kaupunkimittausosasto”.

References

1. Corsar, D., Edwards, P., Baillie, C., Markovic, M., Papangelis, K., Nelson, J.: Get-There: a rural passenger information system utilising linked data & citizen sensing. In: Demo in International Semantic Web Conference 2013, Demo in ISWC 2013, pp. 1–4 (2013). http://www.iswc2013.semanticweb.org/sites/default/files/iswc_demo_22.pdf
2. ECMA-404: The JSON data interchange format (2013). <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Accessed 7 Mar 2017
3. Feng, T., Timmermans, H.J.P.: Transportation mode recognition using GPS and accelerometer data. Transp. Res. Part C Emerg. Technol. **37**, 118–130 (2013). <https://doi.org/10.1016/j.trc.2013.09.014>

³² <http://creativecommons.org/licenses/by/4.0/>.

³³ `device_data`, `device_data_filtered`, `device_models`, `manual_log`.

³⁴ `commuterTrains.json`, `trainStations.json`.

³⁵ `transit_live`.

³⁶ In the subway station entrance images offered in the repository.

4. Heiskala, M., Jokinen, J.P., Tinnilä, M.: Crowdsensing-based transportation services - an analysis from business model and sustainability viewpoints. *Res. Transp. Bus. Manag.* **18**, 38–48 (2016). <https://doi.org/10.1016/j.rtbm.2016.03.006>
5. Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems - SenSys 2013, pp. 1–14 (2013). <http://dl.acm.org/citation.cfm?doid=2517351.2517367>
6. Hertmann, H., Schaefer, C., Kling, C., Janke, D., Niittylä, L., Chatzilari, E.: D1.2 - mobile sensor app with mining functionality. Technical report (2014). <http://cordis.europa.eu/docs/projects/cnect/5/288815/080/deliverables/001-D121.pdf>
7. Kabadyai, S., Pridgen, A., Julien, C.: Virtual sensors: abstracting data from physical sensors. In: Proceedings of 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2006, pp. 587–592. IEEE Computer Society, Washington (2006). <http://dl.acm.org/citation.cfm?id=1139466>
8. Minnigh, P.A., Niittylä, L., de Arana Agirretxa, M., Casasempere, J., Perl, J., Thimm, M., Thiele, F.: End results of trials and Live+Gov methodology. Technical report (2015). <http://cordis.europa.eu/docs/projects/cnect/5/288815/080/deliverables/001-D55.pdf>
9. Nandugudi, A., Ki, T., Nuessle, C., Challen, G.: PocketParker: pocketsourcing parking lot availability. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp 2014 Adjunct, pp. 963–973. ACM Press, New York, September 2014. <http://dl.acm.org/citation.cfm?id=2632048.2632098>
10. Rinne, M.: Towards interoperable traffic data sources. In: 10th ITS European Conference, Helsinki, Finland (2014). https://research.aalto.fi/files/2574973/Towards_interoperable_traffic_data_sources_final.pdf
11. Rinne, M., Törmä, S.: Mobile crowdsensing of parking space using geofencing and activity recognition. In: 10th ITS European Conference, Helsinki, Finland (2014). https://research.aalto.fi/files/2574946/Mobile_crowdsensing_of_parking_space_using_geofencing_and_activity_recognition_final.pdf
12. Shafranovich, Y.: RFC4180: common format and MIME type for Comma-Separated Values (CSV) files (2005). <https://tools.ietf.org/html/rfc4180>. Accessed 7 Mar 2017
13. Shin, D., Aliaga, D., Tunçer, B., Arisona, S.M., Kim, S., Zünd, D., Schmitt, G.: Urban sensing: using smartphones for transportation mode classification. *Comput. Environ. Urban Syst.* **53**, 76–86 (2015). <https://doi.org/10.1016/j.compenvurbsys.2014.07.011>

Guess the Movie - Linking Facebook Pages to IMDb Movies

Paolo Fornacciari¹, Barbara Guidi², Monica Mordonini¹, Jacopo Orlandini¹, Laura Sani¹, and Michele Tomaiuolo¹(✉)

¹ Dipartimento di Ingegneria e Architettura, Università di Parma, Parma, Italy
michele.tomaiuolo@unipr.it

² Dipartimento di Informatica, Università di Pisa, Pisa, Italy

Abstract. In Facebook, the set of pages liked by some users represents an important knowledge about their real life tastes. However, the process of classification, which is already hard when dealing with dozens of classes and genres, is made even more difficult by the very coarse information of Facebook pages. Our work originates from a large dataset of pages liked by users of a Facebook app. To overcome the limitations of multilabel automatic classification of free-form user-generated pages, we acquire data also from IMDb, a large public database about movies. We use it to associate with high accuracy a given cinema-related page on Facebook to the corresponding record on IMDb, which includes plenty of metadata in addition to genres. To this aim, we compare different approaches. The obtained results demonstrate that the highest accuracy is obtained by the combined use of different methods and metrics.

Keywords: Online Social Networks · Data mining
Information Retrieval · Text similarity

1 Introduction

Nowadays, Online Social Networks (OSNs) are the most popular applications in Internet. They changed the way of how people communicate and interact. Structural properties of OSNs and, in particular, understanding users' behavior when they connect to OSNs is crucial for designing user-centered systems. In particular, analysing every aspect of the online social life of users, we could be able to build a kind of system, as a distributed system [9], to model a current Online Social Network, as Facebook. To analyze aspects and characteristics of users, we need information about the online identity of a user, which involves information about the topology of the social graph, friendships and interactions between users, etc. Given several information regarding an online social network user, one point consists in the definition of multi-dimensional networks [4, 14], in which we can mark users with several attributes (i.e. personal interests about cinema, music, etc.). Multidimensionality allows us to analyze each layer separately and at the same time investigate aggregations of different layers. Multi-dimensional

networks can be represented as a multi-graph and due to their complexity, they are difficult to be analyzed. A recent trend in OSNs is to organize the information related to some specific topics (music genre, actors, movies, ...) in pages which can be linked to users through a like operation. This information represents an important knowledge of the real life of users and permits to categorize users according to specific categorizations.

The study of online behaviour of users is a hard task for at least two main motivations: (*i*) The information is very coarse and not easy to categorize because users are free to create what they want and how they want. An example can be done by considering an information related to music pages. In Facebook we can find several pages related to a singer X and several pages dedicated to songs of the singer X. All of these pages should belong to the same category. (*ii*) The lack of a complete dataset which contains information about users, interaction, interests and temporal behaviour (needed to understand the evolution of the social life).

Our work deals with a specific objective, in the scope of these general motivations. In particular, we have tried to overcome some of the mentioned limitations by gathering a real Facebook dataset in which we have all the required information. Thanks to real data, in this paper we provide a tool which can be used to understand some specific personal data of users, with good precision. Indeed, the tool permits to limit the amount of the coarse information obtained from the OSNs by understanding the topic of the information.

We evaluate our tool on Facebook pages concerning movies. The evaluation shows that the tool is able to find a specific film related to a given title and, as a consequence, we are able to understand the genre of the film.

2 Related Works

Traditional methods of Social Network Analysis (SNA, see [21]), are often supported by a semantic analysis of social media to discovery implicit communities that share common tastes and interests in different domains [2]. Community detection is one of the most popular and most computationally expensive application of SNA; it is useful in recommendation systems, link prediction and suggestion, epidemic spreading and information diffusion. An example of community detection algorithm is presented in [1], which adopts an ego-based approach and exploits the parallelism of modern architectures. The problem of discovering topically meaningful communities is introduced in [20]. The framework allows the discovery of both communities and users' interests, with quite good results. In [11] the authors investigate how social networks can be used to generate recommendations and predict the ratings of clubs. The use of both social and content-based information in recommendation system is described, also, in [3]. It presents an inductive learning approach to recommendation, using both ratings information coming from the individuals of a networks and other forms of information about each artifact (such as, cast lists or movie reviews) in predicting user preferences.

Facebook is one of the most popular online social networks and its structure is widely investigated: for example, in [23] the authors describe the social structure obtained by the complete Facebook network of 100 American colleges and universities from a single-day snapshot. Many companies and brands use Facebook to increase their social reputation, or for so-called word-of-mouth marketing. In a recent research [22] the authors investigate the impact of social influence of a Facebook fan page on movie box offices. In fact, almost every movie has its own Facebook fan page nowadays. The research shows that it is possible to establish a relationship between the information diffusion among users on Facebook and the engagements of a fan page without explicit knowledge of the social network, and to measure the social influence of leading users in a fan page. HappyMovie is a Facebook application for recommending movies to groups [18]. The final recommendation is influenced by the personality of each member of the group and the way in which they are connected. This way, the application can offer a product to a group of people that fits the individual needs of every member, trying to achieve high satisfaction. Another interesting Facebook application for a movie recommendation system is presented in [17]. It basically exploits the information coming from Linked Data datasets in the Linked Data cloud, mainly DBpedia and LinkedMDB (the semantic-enabled version of the Internet Movie Database IMDb), to compute a semantic similarity between movies and to provide a recommendation to the user. In [12], authors use both videos and subtitles of movie trailers to classify the genre and MPAA rating.

In the fields of Information Retrieval and document clustering, the possibility of discriminating and selecting documents using text semantic similarity is a quite relevant challenge. In [10], the authors discuss the existing works on text similarity through partitioning them into three approaches: String-based, Corpus-based and Knowledge-based similarities. String-based similarity measures operate on string sequences and character composition. In [5], a variant of the Damerau–Levenshtein Distance is used for spelling corrections. In [13], Cosine Similarity is used for document clustering. Another string matching algorithm is the Gestalt Pattern Matching, or Ratcliff/Obershelp pattern recognition, that is based on the longest common subsequence [19]. An overview of Corpus-based and Knowledge-based similarity algorithms is described in [16]. Corpus-based measures of word semantic similarity try to identify the degree of similarity between words using information exclusively derived from large corpora, while knowledge-based measures try to quantify the degree to which two words are semantically related, using information drawn from semantic networks.

3 Datasets

3.1 The SocialCircles! Dataset

We use a real dataset gathered by a Facebook application, called SocialCircles!¹, which exploited the Facebook API to retrieve social information about registered

¹ <http://social.di.unipi.it/>.

users. The application is now unavailable because Facebook no longer support the applications exploiting this API since the 1st May 2015. The goal of the application was to retrieve a good number of Ego Networks, which are explained in detail in [8]. Instead our specific scenario is explained in [7] and is associated with the following sets of data:

- Topology and profile information. We gathered complete profile information about 337 registered users and the friendship relations existing between them.
- Interaction information. We collected information about interactions between users registered to the application and their friends, such as posts, comments, likes, tags and photos.
- Online presence data. We monitored the chat presence status and obtained information about the time spent online by registered users and their friends, for a total of 95.578 users.

The dataset obtained from the SocialCircles! application contains 337 complete Ego Networks, for a total of 144.481 users (ego and their alters). The Ego Networks we retrieved have the advantage of representing a very heterogeneous population: 213 males and 115 females, with age range of 15–79 with different education, background and provenience. We believe this is an advantage compared to other works where the dataset comes from an analysis of groups of people with the same background and education [15]. In [6] a complete analysis of the dataset is shown.

3.2 Multi-layer Ego Network: Users, Interactions, and Movies

As explained before, we are able to build a multi-layer network which contains different layers, one to describe the ego network of a generic ego, one layer to describe the interaction between users in an ego network and, finally, one layer to describe an interest of users, in our case we focus on movie genres. This multi-dimensional ego networks could be extended in the future with other information contained in our dataset, such as music, books, etc., which can be used to categorize users, find homophily and, to recommend information based on user interest. As shown in Fig. 1, from the SocialCircle dataset we extract information about 69519 Facebook movie pages and the relations between users and the movie pages (who likes what). Several of them are pages without a significant title and the most important problem when we need to tag a user with a genre is to understand the genre of a page from the title of the page, that is the only information available. Thus, for each page we also collect all available posts and comments.

3.3 The IMDb Dataset

The Internet Movie Database (IMDb)² is an online relational database of information related to movies, television programs and video games, including cast,

² <http://www.imdb.com/>.

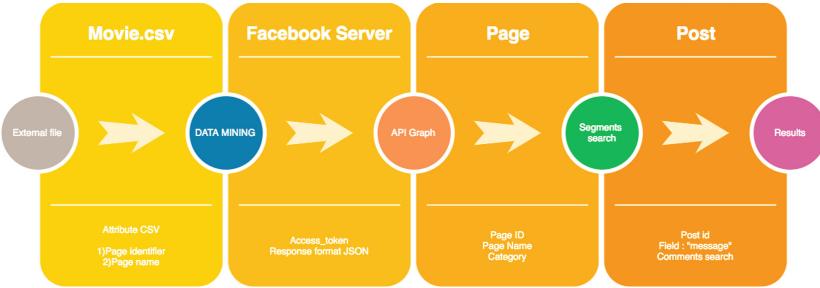


Fig. 1. Workflow of data acquisition from Facebook, for a given public page.

production crew, fictional characters, biographies, plot summaries and reviews, handled by IMDb.com, Inc., a subsidiary of Amazon. IMDb contains the Top Rated Movie list, ranked by a formula which includes the number of ratings each movie received from users, and the value of ratings received from regular users. To be included on the list, a movie must receive ratings from at least 25000 users. Using the same threshold for popularity, the dataset used in this work is made up of all the 4636 unique films that received ratings from at least 25000 users on IMDb. In addition to the title, the dataset contains synopsis, plot and genre. Each movie can be associated to one (or more) of the 27 genres. The textual elements analyzed in our work are titles, plots and synopsis.

4 Methodology

In our system, the main algorithm confronts a given Facebook page with a list of IMDb movies, to find the best matching movie. Algorithm 1 is essentially a function to select the movie with the maximum similarity, with respect to the given page. The parameters represent, respectively, the given page (p) and the set of movies (\mathcal{M} , in our case, the whole dataset acquired from IMDb). Both the *SelectMovies* function and the *PageSimilarity* function are customizable in different ways. In fact, we experimented different mechanisms and policies, to finally find the most suitable combination.

In all our experiments, but the very first, the *SelectMovies* function is an implementation of Algorithm 2. Each IMDb movie is associated with its title similarity, with respect to the given Facebook page. Thus, a list of pairs (*similarity, movie*) is obtained. This list is then sorted and its first n elements are considered for further analysis. Here, n is the result of an implementation of the *SliceSize* function, which we will discuss later.

4.1 Approach #1: Content Similarity

Since titles are not accurate, the first approach we have experimented is simply to discard them completely. This analysis is based on the content of Facebook

Algorithm 1. Pseudo-code of the main function.

```

1: function FINDMOVIE( $p, \mathcal{M}$ )
2:    $\mathcal{L} \leftarrow SelectMovies(p, \mathcal{M})$ 
3:    $s_{best} \leftarrow 0$ 
4:    $m_{best} \leftarrow None$ 
5:   for all  $ts, m \in \mathcal{L}$  do
6:      $s \leftarrow PageSimilarity(p, m, ts)$ 
7:     if  $s > s_{best}$  then
8:        $s_{best} \leftarrow s$ 
9:        $m_{best} \leftarrow m$ 
10:    end if
11:   end for
12:   return  $m_{best}$ 
13: end function

```

Algorithm 2. Pseudo-code of the selection of movies.

```

1: function SELECTMOVIES( $p, \mathcal{M}$ )
2:    $\mathcal{L} \leftarrow \emptyset$ 
3:   for all  $m \in \mathcal{M}$  do
4:      $ts \leftarrow TitleSimilarity(p, m)$ 
5:      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(ts, m)\}$ 
6:   end for
7:    $\mathcal{L} \leftarrow Sort(\mathcal{L})$ 
8:    $n \leftarrow SliceSize(\mathcal{L})$ 
9:    $\mathcal{L} \leftarrow Take(n, \mathcal{L})$ 
10:  return  $\mathcal{L}$ 
11: end function

```

pages and IMDb movies, without considering their titles. Given a certain page p , we confront it with each movie m in our database, based on their textual content. For the Facebook page, the text is collected from all posts and comments on the page. For the IMDb movie, the text is obtained by concatenating its synopsis (if it is available) with all its plots.

Referring to Algorithm 1, the *SelectMovies* function is essentially replaced by an all-pass function. All available movies are accepted and assigned the same title distance. The *PageSimilarity* function is evaluated as the *Cosine Similarity* between the page p and the movie m . As we will describe later, this approach is very computational intensive, as it compares a given Facebook page with every available movie. Moreover, its accuracy is not satisfactory.

Cosine Similarity has been used as it is one of the best known and most precise algorithms for evaluating the similarity between two long texts. It uses the classical BagOfWords transformation, for representing a text in a multidimensional space. Features are then weighted according to the TfIdf model, which highlights the most peculiar words of a given text, instead of the most frequent words. Texts in our datasets are all prefiltered, for removing stop words, and stemmed.

4.2 Approach #2: Variable Slice

The second approach uses the similarity between titles, in order to obtain a list of the most promising movies. This has the aim of both reducing calculations and also guiding the process towards the most interesting set of movies.

For confronting titles, we have initially considered the Levenshtein distance and the Gestalt Pattern Matching (GPM) function. However, from the initial results, the Levenshtein distance has demonstrated to be unfit for this kind of analysis. Thus, for implementing the *TitleSimilarity* function, we have used the Gestalt approach for pattern matching.

For deciding the size of the slice of movies to consider, we have studied the data in our annotated dataset. In particular, we have plotted a set of points for these data, in a Cartesian space, where the x-axis represents the obtained best value of similarity (not necessarily for the correct movie) and the y-axis represent the position of the correct movie in the sorted list. A linear regression gives better clue about the most useful region of this space. In fact, when the best obtained similarity is low, the position of the correct movie tends to be higher. Thus, in our system, the number of movies considered for full textual analysis is a variable slice. The implementation of the *SliceSize* function closely approximates this regression line, i.e., the slice size is inversely proportional to the best obtained similarity (which may not correspond to the correct movie).

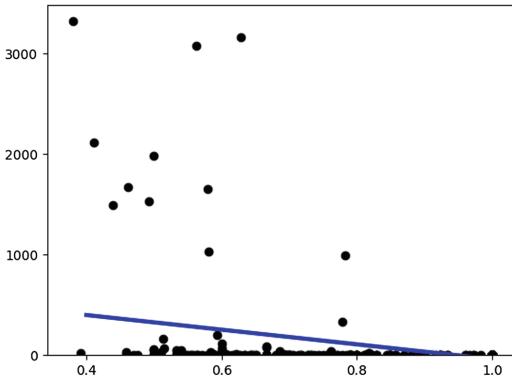


Fig. 2. Linear regression applied to sample data. Each dot represents an instance of the sample dataset, as a pair ($similarity_{best}$, $position_{correct}$). I.e., we have matched each obtained best value of similarity (which may not correspond to the correct movie), with the position of the correct movie in the sorted list. The line is the linear regression.

4.3 Approach #3: Gestalt Title Similarity

As an alternative evaluation, instead of measuring the Cosine Distance between textual content, we use only the title similarity (based on Gestalt Pattern Matching). In Algorithm 2, this corresponds to a constant slice size of a single element: the movie with the best matching title.

4.4 Approach #4: Levenshtein Title Similarity

For direct comparison, we repeat the previous approach, using a measure of similarity based on the Levenshtein Distance, instead of the Gestalt Pattern Matching. The similarity measure is essentially obtained from the inverse of the distance. In the following formula, x is the Levenshtein Distance and y the similarity (k is a parameter to optimize).

$$y = \frac{1}{1 + k \cdot x}$$

4.5 Approach #5: Title + Popularity

Additionally, for selecting the most promising movies, their popularity may also be useful. Thus, we evaluate a popularity index, on the basis of votes collected on IMDb. We consider the total number of votes, disrespecting their particular value. Either voted positively or negatively, a movie has to be considered popular or known, if it collects a large quantity of feedback. For evaluating the popularity index, we use the following function, which penalizes the less popular movies (h is a parameter to optimize). It is worth to remember here that we have collected only IMDb movies with at least 25000 votes. The title similarity is then combined linearly with this popularity index. Popularity is weighted with a small coefficient, as it is useful to discriminate only among very similar titles.

$$y = \left(1 - \frac{25000}{x}\right)^h$$

4.6 Approach #6: Title + Popularity + Content

Finally, according to this last approach, both the textual content and the title provide useful information for finding the right movie. Thus, in this case the *PageSimilarity* returns a linear combination of title similarity (Gestalt Pattern Matching) and content similarity (Cosine Similarity).

Cosine Similarity is inferior, alone. But it provides a good contribution, if joined with title similarity.

5 Results

We have tested all the approaches described in Sect. 4 on a subset of Facebook pages, from the global dataset. These randomly selected pages have been manually annotated and accurately matched with their corresponding IMDb movies³. This testset includes 327 pages, which are of very different kinds, including some titles with misspelled words, cryptic citations, names of actors and characters instead of the movie title, non-English language, etc.

³ The main datasets of this work are available at <http://sowide.unipr.it/datasets>.

Table 1. Results of the different approaches, for both quality and computing time.

Approach	Accuracy	Precision	Recall	F-measure	Time
#1	14.98%	0.1887	0.1503	0.1596	22611.52 s
#2	68.50%	0.8021	0.6840	0.7095	267.18 s
#3	83.79%	0.8824	0.8374	0.8439	70.09 s
#4	81.65%	0.8901	0.8160	0.8279	7.46 s
#5	84.10%	0.8870	0.8405	0.8445	75.92 s
#6	86.24% (93.88%)	0.8944	0.8620	0.8666	314.26 s

In particular, for the content similarity, we always prefilter the textual content for removing English classical stop words and stemming the remaining words; for title similarity, we remove only the words “the” and “movie”, which are often used in names of Facebook pages and largely disturb the analysis, and then remove all spaces. In all approaches, we use the following optimizations: the slice size decreases from 720 at $r_{best} = 0$, to 1 at $r_{best} \geq 0.9$; for the Levenshtein similarity, k is 0.05; for the popularity index, h is 2. In all our tests, the Levenshtein similarity has always provided lower accuracy than the Gestalt similarity, though executing in much less time.

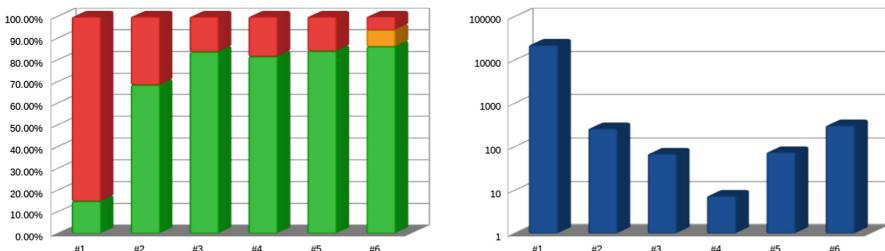


Fig. 3. Accuracy (left) and execution time (right) of the six different approaches described in Sect. 4. For the accuracy, the last bar shows also the quota of wrong movies, but still in the correct series/saga. For the execution time, the results are in seconds and the scale is logarithmic.

We have found that in many cases the correct movie can be determined simply through the comparison of titles (approach #5). However, for obtaining the correct result when the page title is misleading, the comparison of page content and movie description is needed (approach #2). The final result (approach #6) is obtained by combining title and content similarity, weighted at 0.40 and 0.60, respectively; weights are optimized, meaning that content similarity gives an important contribution. Results are shown in Table 1 and Fig. 3. We list here few anecdotal examples (when a FB page is dedicated to a whole saga, in the test set it is linked to the first episode in IMDb).

- FB page: ... *Twilight.*, *New Monn.*, *Eclipse and Breaking dawn...* [typo in the title], correctly classified as: *The Twilight Saga: New Moon*;
- FB page: ~*TwILiGhT~neW MoOn ~eCLipse~bReakiNg DAwN~*, correctly classified as: *The Twilight Saga: New Moon*;
- FB page: *noi...twilighters forever...*, correctly classified as: *The Twilight Saga: New Moon*;
- FB page: *Toothless the unique NightFury dragon*, correctly classified as: *How to Train Your Dragon*;
- FB page: “*So that’s who Finnick loves. A poor, mad girl back home.*” [quotes in the title], correctly classified as: *The Hunger Games*;
- FB page: *New Zealand Hobbit Fans*, correctly classified as: *The Hobbit: An Unexpected Journey*;
- FB page: *Meet the Parents: Little Fockers*, wrongly classified as: *Meet the Fockers*, instead of: *Little Fockers*;
- FB page: *Matrix Trilogy*, wrongly classified as: *The Matrix Reloaded*, instead of: *The Matrix*;
- FB page: “*boom boom!...fire powerr!*” (*Night At the Museum 2*) [quotes in the title], wrongly classified as: *Night at the Museum*, instead of: *Night at the Museum: Battle of the Smithsonian*.

Finally, by a closer look at the errors, it can be easily observed that many wrong movies are still selected from the correct series/saga. In many cases this might still be a useful result. Counting also these instances as acceptable, the final accuracy would be at 93.88%. In our opinion, the final result is pretty good also for real world application. It is mainly due to the inclusion of external data, IMDb in this case, in the process. These data add valuable knowledge, which is exploited by the prediction system. As a downside, this kind of approach cannot deal with pages related to less known or too recent movies, which are not present in the IMDb list of popular movies, by number of votes.

In the end, it is worth to consider that a bayesian multi-label classifier (27 labels) has shown an accuracy of 68.38%, in predicting *any one* of the genres associated with a movie. This classifier has been trained and tested, with 10-folds cross validation, on the synopsis and plots from IMDB, which usually are more coherent than Facebook pages. In [12], using movie subtitles, the genre is classified with an accuracy of 43.66%, using a Support Vector Machine classifier, with 7 possible output labels.

6 Conclusions

The motivation for this research work was to study the possibility of inferring some Facebook users’ tastes about cinema genres, by analysing the pages they like. Multi-class automatic classification with high accuracy, as required in the case of the dozens of movies genres, is always challenging. It is even more so in case of coarse user-generated pages. Our approach is based on the use of IMDb data, to associate a Facebook page with an IMDb record, instead of direct genre

classification. This way, on the basis of title and content similarity, we are able to obtain a high level of accuracy in finding the exact movie which the Facebook page is dedicated to. Thus, we gain not only genre information, but also all the metadata in IMDb.

References

1. Moretti, M., Ferrari, A., Fornacciari, P., Mordonini, M., Rosi, F., Tomaiuolo, M.: Local-first algorithms for community detection. In: Proceedings of the 2nd International Workshop on Knowledge Discovery on the WEB, KDWeb 2016, Cagliari, Italy, 8–10 September (2016). <http://ceur-ws.org/Vol-1748/paper-07.pdf>
2. Angiani, G., Fornacciari, P., Iotti, E., Mordonini, M., Tomaiuolo, M.: Models of participation in social networks. In: Social Media Performance Evaluation and Success Measurements, p. 196 (2016)
3. Basu, C., Hirsh, H., Cohen, W.: Recommendation as classification: using social and content-based information in recommendation. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, AAAI 1998/IAAI 1998, pp. 714–720. American Association for Artificial Intelligence, Menlo Park (1998)
4. Boccaletti, S., Bianconi, G., Criado, R., Del Genio, C.I., Gómez-Gardenes, J., Romance, M., Sendina-Nadal, I., Wang, Z., Zanin, M.: The structure and dynamics of multilayer networks. *Phys. Rep.* **544**(1), 1–122 (2014)
5. Cucerzan, S., Brill, E.: Spelling correction as an iterative process that exploits the collective knowledge of web users. In: EMNLP 2004, pp. 293–300 (2004)
6. De Salve, A., Dondio, M., Guidi, B., Ricci, L.: The impact of user's availability on on-line ego networks. *Comput. Commun.* **73**(PB), 211–218 (2016)
7. De Salve, A., Guidi, B., Mori, P., Ricci, L.: Distributed coverage of ego networks in F2F online social networks. In: 2016 International IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), pp. 423–431. IEEE (2016)
8. Everett, M., Borgatti, S.P.: Ego network betweenness. *Soc. Netw.* **27**(1), 31–38 (2005)
9. Franchi, E., Poggi, A., Tomaiuolo, M.: Blogracy: a peer-to-peer social network. *Int. J. Distrib. Syst. Technol. (IJDST)* **7**(2), 37–56 (2016)
10. Gomaa, W.H., Fahmy, A.A.: A survey of text similarity approaches. *Int. J. Comput. Appl.* **68**(13), 13–18 (2013)
11. Groh, G., Ehmig, C.: Recommendations in taste related domains: collaborative filtering vs. social filtering. In: Proceedings of the 2007 International ACM Conference on Supporting Group Work, pp. 127–136. ACM (2007)
12. Helmer, E., Ji, Q.: Film classification by trailer features (2012)
13. Huang, A.: Similarity measures for text document clustering. In: Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC 2008), Christchurch, New Zealand, pp. 49–56 (2008)
14. Kazienko, P., Musial, K., Kukla, E., Kajdanowicz, T., Bródka, P.: Multidimensional social network: model and analysis. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011. LNCS (LNAI), vol. 6922, pp. 378–387. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23935-9_37

15. La Gala, M., Arnaboldi, V., Conti, M., Passarella, A.: Ego-net digger: a new way to study ego networks in online social networks. In: First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research (ACM HotSocial 2012) (2012)
16. Mihalcea, R., Corley, C., Strapparava, C., et al.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI 2006, pp. 775–780 (2006)
17. Mirizzi, R., Di Noia, T., Ragone, A., Ostuni, V.C., Di Sciascio, E.: Movie recommendation with dbpedia. In: IIR, pp. 101–112 (2012)
18. Quijano-Sanchez, L., Recio-Garcia, J.A., Diaz-Agudo, B.: HappyMovie: a Facebook application for recommending movies to groups. In: 2011 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 239–244. IEEE (2011)
19. Ratcliff, J.W., Metzener, D.E.: Pattern-matching - the Gestalt approach. Dr. Dobbs J. **13**(7), 46 (1988)
20. Sachan, M., Contractor, D., Faruquie, T.A., Subramaniam, L.V.: Using content and interactions for discovering communities in social networks. In: Proceedings of the 21st International Conference on World Wide Web, pp. 331–340. ACM (2012)
21. Scott, J.: Social Network Analysis. Sage, Thousand Oaks (2017)
22. Tang, W.H., Yeh, M.Y., Lee, A.J.: Information diffusion among users on Facebook fan pages over time: its impact on movie box office. In: 2014 International Conference on Data Science and Advanced Analytics (DSAA), pp. 340–346. IEEE (2014)
23. Traud, A.L., Mucha, P.J., Porter, M.A.: Social structure of Facebook networks. Phys. A Stat. Mech. Appl. **391**(16), 4165–4180 (2012)

Research on Online Digital Cultures - Community Extraction and Analysis by Markov and k-Means Clustering

Giles Greenway^(✉), Tobias Blanke, Mark Cote, and Jennifer Pybus

Department of Digital Humanities, King's College London, London, UK
giles.greenway@kcl.ac.uk

Abstract. We investigate approaches to personal data analytics that involves the participation of all actors in our shared digital culture. We analyse their communities by identifying and clustering social relations using mobile and social media data. The work is part of our effort to develop tools to create a “social data commons”, an open research environment that will share innovative tools and data sets to researchers interested in accessing the data that surrounds the production and circulation of digital culture and their actors. This experiment focuses on the groups of clustered relations that are formed within a user’s social data traces. Community extraction is a popular part of the analysis of social data. We have applied the technique of Markov Clustering to the Twitter networks of social actors. Qualitatively, we demonstrate that it is more effective than the Louvain method for finding social groups known to the subjects, while still being very simple to implement. We also demonstrate that traces of cell towers captured using our “MobileMiner” mobile application are sufficient to capture significant details about their social relations by the simple application of k-means.

1 Introduction

Applications, especially social media applications, on mobile devices are frequently given access to information about their users’ location, through information about cellular network towers, wireless networks and full GPS access. When this is combined with the media or messages the users choose to send or receive, an application can learn a great deal about them. This data is often used to target advertising or recommend other content or connections with other users. Some studies are able to negotiate with mobile service operators for the use of customers cell tower data, under certain conditions. Others have made use of mobile application data through agreement with social media companies or use of their APIs within terms-of-service. Levandoski *et al.* [1] collected data from the Foursquare service using its public API, but a page on the University of Minnesota website [2] states that this is no longer available at Foursquare’s request. Since this data is anonymised, it is not possible to use it in discussions with mobile device users about their attitudes to data and privacy.

One way to secure such data for long-term use without conditions is to collect it from end-users with their full informed consent and participation. To this end, we have collaborated with 20 young programmers recruited via Young Rewired State [3] (YRS). An Android application provisionally called MobileMiner [4] was developed that logs data commonly gathered by other mobile applications, as well as their network activity. The application was developed in public, with the source-code made available under version 2 of the Apache license so the YRS members could be aware of its activities and participate in its development. They were issued with smartphones with the application installed, and used it to log data over 5 months. During this time, we held two hackday events attended by the YRS members. The first had a theme of application development, the other geared towards returning the data to the participants so they could analyse it for themselves [5]. We have already reported on MobileMiner's collection of network socket data [4]. Here, we investigate the insights that can be gained by its capture of the mobile cell towers user's devices connect to. In order to reason about what insights third-party app developers could gain from this data, given they also have the users' generated content, we also analysed the Twitter networks of the YRS volunteers.

The exponential growth of both transactional data and metadata that is generated within social media platforms raises questions in relation to how data is archived and assigned value [6, 7]. Commercial services such as Conversocial [8] and SproutSocial [9] provide the means for businesses to analyse the perception of their brands in social media, in order to manage customer complaints and capitalize on positive reviews. Others, like Twitonomy [10] and Simply Measured [11] track the effectiveness of social media campaigns (how far a Tweet is retweeted, for instance) and identifying key influencers. The TrendMiner [12] project also enables the mining of social media data, with the stated goals of enabling political analysis and economic decision-making.

The focus of these tools is seldom directed towards users and how they use social media in their everyday lives. We believe that there is a need for tools that allow social media users to gain awareness of how the data they generate are being used. In light of analysis of this data, users may assess how they might want to change their use of such platforms. To this end, we provide a "Social Data Commons" in the "Our Data Ourselves" project [13]. This kind of work is often dominated by commercial interests who treat the content produced by users as the central point of value in exchange for "free" online services.

Twitter has demonstrated to be a rich source for network analysis and communication patterns. It allows us to investigate, for instance, the degree to which users are consumers of content from a few accounts with many followers or whether they use the platform conversationally between other users with reciprocated follower relationships. The fraction of these relationships which have involved sustained conversations and the frequency distributions of their lengths has also been studied [14]. Lastly, the degree to which a user's friends and followers network can be segmented into communities or clusters is of key interest. Members of such clusters will have more friend/follower relationships between

them than with those in other groups. Himmelboim *et al.* [15] used NodeXL spreadsheet tool [16] to retrieve tweets containing certain keywords, then cluster the accounts that tweeted them into communities. It was observed that there were far more interactions within communities than between them.

Our experiment uses the Markov Clustering (MCL) graph-clustering algorithm, [17] which is very simple to implement. We apply this algorithm to analyse how sociality develops within the networked relationships around a Twitter account.

2 The MobileMiner App

The Android operating system was pragmatically chosen as the mobile platform for the project; it has the largest share of the smartphone market and requires less investment to release code and apps. Following our co-development approach with YRS, it was essential that the YRS members would be able to contribute to app development. The Apple IOS, for instance, limits these options significantly. The tool-chain for development on Apple's IOS is only available for its MacOS operating system. MacOS can be run on virtual machines without access to Apple hardware, but not without breaching its end-user license agreement. In order for the YRS members to send and receive updates to MobileMiner and install it on their devices, all would require access to Apple hardware. The only other way to distribute the app would be via Apple's I-tunes store, which would have been subject to its approval. In contrast, the Android operating system gives the user the option of installing software from sources other than Google's Play store. The Android development tool-chain or software development kit (SDK) is cross-platform, and released under the Apache open-source license, albeit with restrictions on the distribution of the supplied binary executable files.

The MobileMiner Android app gives users the option to start and stop recording data whenever they choose. It displays the active mobile or wireless network, the id of the current cell tower, and recent app activity in terms of network socket usage (Fig. 1). Newly gathered data is written to a SQLite database on the device at 5 min intervals. Storing each item as it arrived would cause very frequent writing to the device's flash storage, and have a very severe effect on battery life. To avoid accruing mobile data costs to the user, new data is uploaded by an http request to a server every 10 min only if the device has a connection to a wireless network. There are some basic features to examine the collected data on the device. A list of apps ordered by the number of network socket events is provided, and the users may also request a heat-map showing how frequently they visit the cell towers known to the OpenCellID [18] database. The maps are drawn using OpenStreetMaps [19] via the OpenLayers JavaScript library. Users have the option to copy the app's SQLite database to an accessible part of their device's flash storage. Users do not have access to the databases of the apps on their devices by default; root access and the SDK are required. The YRS members were encouraged to access their data using the SQLite module

in the Python standard library during the first hack-day, although SQLite apps are available for examining the data on the device.

Android apps may request GPS location data from two sources, either from the device's sensor or via Google's Play Services API. The latter is combined with location data from cell towers and wireless networks, but requires agreeing with Google's terms and conditions, as does the display of Google's Maps within an app. The precision afforded by such data could allow individuals to be identified by their home address. Frequent and repeated use of the GPS sensor would also significantly reduce the device's battery life. It is therefore more attractive to use the locations of cell towers as a proxy for approximate location. Apps using Android's 'coarse location' permission are fed location information based on cell towers and wireless networks, but short of disabling wireless connections on the device, there is no way to limit this to cell towers only. The extra precision of location afforded to the coarse location API by adding wireless network information is too high for it to be captured by the app. MobileMiner uses the Android API to collect the IDs of cell towers as the device connects to them, then finds their locations using data from the OpenCellID database. This contains crowd-sourced measurements of the locations of cell towers, combined with known locations from some mobile network operators that freely publish this information. Unlike collecting cell IDs only during call or SMS events from the logs of mobile operators, [20] these trails of locations are continuous as long as the app is in use, subject to the sparseness of the OpenCellID database.

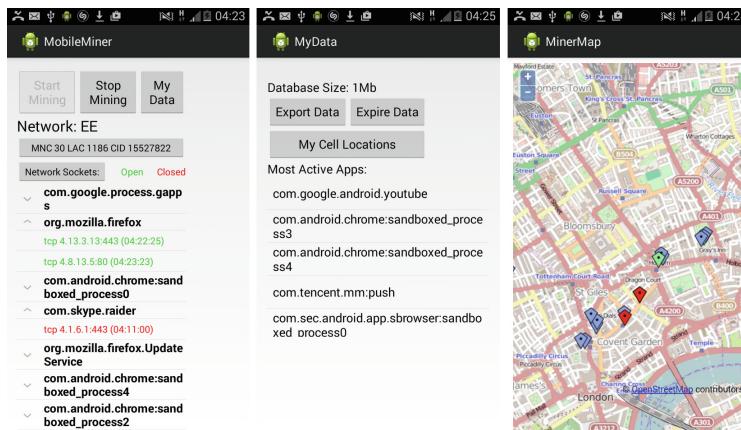


Fig. 1. Screenshots of the MobileMiner app, showing network sockets, apps ordered by total activity, and a heat-map of cell towers.

3 MCL

MCL community detection works by reasoning about random walks on a network, for which an adjacency matrix is created. The columns are normalized,

such that entries for each column node represent the chance of visiting a connected node chosen at random. Squaring the matrix produces columns of probabilities of node occupancy after two such random steps, starting from the column node; this stage is called expansion. The elements of the matrix are then raised to a power called the inflation parameter (2 in this case) and the columns are renormalized. This has the effect of raising higher probabilities and reducing lower ones. The two steps are repeated until the matrix converges, when most of the entries will be close to 0 or 1. The non-zero entries in a row will represent probable starting nodes for a random walk, given that it ended in the row node. Assuming that a random walk starting in a densely connected community of nodes is more likely to remain within it than to leave it, the sets of nodes with non-zero entries for each row represent clusters.

3.1 Acquiring Twitter Data

Before applying MCL to the Twitter accounts of the YRS volunteers, we tested it using that of a digital culture researcher, also an author of this paper. User profiles were extracted with the Twitter API and stored in the Neo4J graph database. The properties returned by the API call were used to form the attributes of each node representing a user. The users' friends and followers were retrieved by repeated API calls; these relationships were modeled by creating directed edges between the nodes in the database. Having seeded the users in the Neo4J graph, the friends and followers were expanded in the following way, using Neo4J's graph query language Cypher:

1. Find set of user nodes who have a friendship relationship with the seed user in any direction.
2. Find the set of user nodes who have a relationship with any users in the previous set.
3. Find the count of outgoing (friends) or incoming (follower) relationships already within the database for each node in this set.
4. Compare the count to the expected number of relationships as described by the “friends_count” or “followers_count” attributes returned by the Twitter API calls.
5. Return the users for which the count is lower, in ascending order of when they were retrieved.
6. Expand these users friend-follower relationships and repeat.

3.2 Applying and Evaluating MCL

A set of Twitter users with bi-directional follower relationships was constructed with another query in order to be analysed with MCL, implemented using NumPy [21]. All such relationships were returned for users within two relationships of the target user. An adjacency matrix was constructed to represent this network, with the diagonal filled with 1s to add a self-loop to each node. Having normalized the columns, the matrix was squared, raised to the power of

an inflation parameter of 2.0 element-wise, and renormalized. The process was repeated until the standard deviation of the absolute differences between successive matrices was less than 10^{-3} . Clusters were extracted by considering rows with more than three non-zero elements. For the first Twitter account examined, the graph matrices was around 7000×7000 in size, with the algorithm converging in around 25 iterations.

We chose to evaluate our implementation of MCL by comparing it with an out-of-the-box approach and then let the researcher evaluate the results and label them. To this end, we used the popular graph analysis tool Gephi [22], which implements the Louvain clustering algorithm [23]. This algorithm involves successively joining clusters together such that modularity, which measures the density to intra-cluster links to inter-cluster ones, is maximized. The researcher then rated each of their clusters returned on a scale of 1–3, where 1 indicated that the individuals within the cluster were connected by a clearly identifiable common denominator and research goal, 2 that the individuals seemed connected, but the context was not clear, and 3 that the individuals seemed to have no connection. The researcher was not involved with the collection of data or the implementation of MCL, and was unaware of which methods were used to generate the clusters. This semi-automated approach has proven effective for the analysis of clusters [24].

The Twitter account had 7144 others within two reciprocal friend/follower relationships. The MCL algorithm placed 55% of these in clusters with more than three members, the remainder were considered unclustered. Just over 8% of the accounts were in the largest cluster, the next largest had 5%, 5% and 4% of users respectively. This tendency towards large numbers of small clusters suggests that MCL may be over-fitting by partitioning larger clusters into smaller ones. The clusterings are summarized in Table 1.

At the time of capturing the data, the account had 319 friends and 346 followers, with 100 users in both groups. The qualitative cluster labeling exercise resulted in 20% of clusters being given the highest grade of relevancy, these contained 195 Twitter accounts. 37% of clusters received the second highest grade. Given that the vast majority of the Twitter accounts in the network were unknown to the target user, the fraction of clusters with no apparent relevance does not seem unduly high. MCL has clearly been able to extract some meaning from the account's connections, with many clusters clearly recognizable as belonging to specific institutions or conferences. All of the clusters obtained by the Louvain method received the lowest grade of relevance. Qualitatively, MCL clearly extracts meaningful communities as observed by the owner of the Twitter account surveyed, and has the advantage of clarity of application. There is no definitive quantitative definition for graph clusters [25], nor a definitive quantitative method for evaluating them [26].

3.3 Results

The same Twitter-crawling algorithm was applied to the 9 YRS volunteers with Twitter accounts, and MCL and the Louvain method were applied to

Table 1. Fractions of clusters achieving each relevancy score, for MCL and the Louvain method.

	MCL	Louvain
1. Cluster is identifiable and relevant	20%	0%
2. Cluster is not identifiable, but possibly relevant	37%	0%
3. Cluster is neither identifiable nor relevant	43%	100%

all the accounts collected. MCL returned 13 clusters, with all but one of the YRS accounts placed in the largest, with 319 accounts as shown in Table 2. The remaining YRS member was placed in the second largest cluster, with 45 accounts. The Louvain method failed to identify the YRS members as a coherent group, and distributed them across four clusters with 1–3 members in each. If the Twitter communities detected by MCL are to have true social context, it is reasonable to expect them to Tweet about well-defined topics. This is clearly the case, as shown by the hashtags used in the MCL clusters containing the accounts of YRS members in Table 3. Both clusters are dominated by the 2015 UK general election, but the cluster with more YRS members predominately uses tags about the YRS organization and technology. The smaller cluster is mostly concerned with UK politics.

Table 2. Distribution of YRS member Twitter accounts in clusterings by MCL and the Louvain method.

MCL	cluster size	20	26	6	6	5	45	5	319	6	5	14	14	5
	YRS accounts	0	0	0	0	0	1	0	8	0	0	0	0	0
Louvain	cluster size	15	78	7	43	168	67	55	230	24				
	YRS accounts	0	1	0	0	0	3	2	3	0				

4 Mobile Cell Tower Data

When MobileMiner is in use, it creates a time-series of cell tower IDs, which using the standard Android Java API. Many of the captured cell towers were known to the OpenCellID database. We explore how much information these trails reveal about the users. One of the most prolific YRS users of the app visited and logged 930 unique cell towers in 148 days. The latitudes and longitudes of 253 (27%) of these were known to OpenCellID.

A simple approach to analysing user behaviour is to cluster the cell towers spatially, [20] and then examine the times of occupancy of the clusters. k-Means clustering is best suited to convex clusters in a low-dimensional Euclidean space, these conditions are met by the cell-tower data reasonably well. The k-means

Table 3. Hashtag usage for MCL clusters containing YRS members.

8 YRS accounts		1 YRS account	
Hashtag	Tweets	Hashtag	Tweets
GE2015	275	GE2015	78
tech	214	Eurovision	58
jobs	207	leadersdebate	33
YRS2014	185	bdw2015	24
Haunted	183	BattleForNumber10	21
ghosts	183	BBCQQT	18
YRSFoc	181	GBR	15
hackmcr	167	bbcqt	14
yrs2014	156	eurovision	14
Arduino	149	NHTG15	13
FoC2015	141	FoC2015	12
Norwich	133	YRSAmbassadors	11
gamedev	132	depop	11
TG	130	BBCFreeSpeech	10
BigData	112	VoteConverative	9
linux	111	YRS2014	9
YRSHyperlocal	105	DimblebyLecture	9
design	99	endpointcon	9

routine from the SciKitLearn Python package [27] was applied to the normalized location data for increasing values of k clusters, until the mean distances of the points to their assigned cluster centres stopped decreasing appreciably. In this case iteration stopped when the mean distance was greater than an arbitrary fraction of 90% of the previous lowest value.

Figure 2 shows the clustering using simple feature vectors consisting only of latitude and longitude. It is attractive to capture journeys, as well as places. Extended groups of points near the centre of the distribution are split into three clusters. It is likely that these are part of a single journey, with the breaks between the clusters being accounted for by poor reception of the mobile network. To mitigate this, the changes in latitude and longitude between successive points were divided by the time intervals between them to estimate the velocity at each point. The clustering was repeated using feature vectors consisting of both position and velocity, as shown in Fig. 3. Points that are part of the same car or train journey will have similar velocities, and can therefore be assigned to the same cluster. A mobile device might connect to a new cell tower because of network traffic or temporary loss of reception, rather than any change in proximity, so this measure of velocity cannot be expected to be realistic. Such random fluctuations should tend to cancel for clusters consisting of groups of cell

towers that corresponded to places rather than journeys. These remained stable for both clusterings.

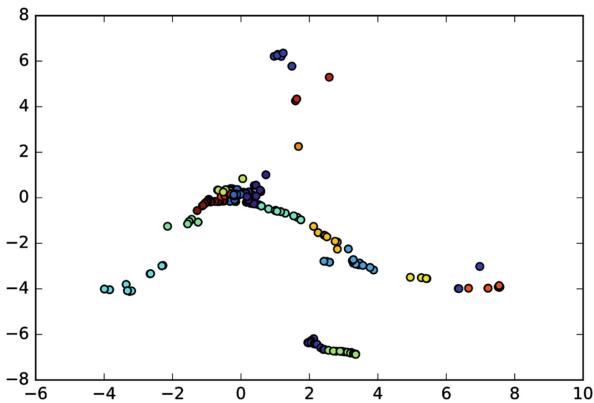


Fig. 2. Cell tower locations with normalized latitude and longitude for a single user, clustered by K-means using positions as feature vectors.

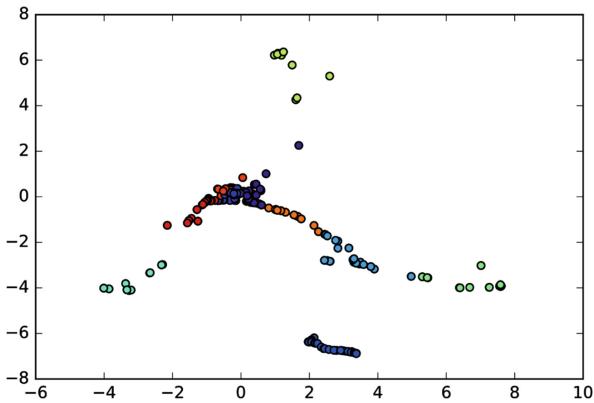


Fig. 3. Cell tower locations with normalized lattitude and longitude for a single user, clustered by K-means using positions and velocities as feature vectors.

4.1 Temporal Analysis

The sequences of known cell tower locations for each user were clustered by k-means using positions and estimated velocities as feature vectors and each cell tower in a sequence was labeled with its assigned cluster. For each cluster,

the number of days between its first and last visit and the number of days with at least one cell tower connection were determined. The total number of measurements in the cluster on weekdays and weekends were multiplied by $\frac{7}{5}$ and $\frac{7}{2}$ respectively. If the weekday product was greater than 5 times the weekend product, the cluster was deemed to have been mostly visited during weekdays, if the weekend product was greater than 5 times the weekday product, the cluster was considered to have been mostly visited during weekends. The histograms of the hour of each cell tower connection in each cluster were found. The earliest and latest hours where the number of measurements was at least one fifth of the maximum values were used to estimate the range of time when each cluster tended to be occupied. The name of each cluster's location was found by querying OpenStreetMaps [19] API with the latitude and longitude of its centre.

For the user whose data was plotted in the previous figures, this approach yielded a cluster active during all hours of the day for all days of the week, that correctly corresponded to the district where they lived. Another, active during weekdays from 09:00 to 19:00, plausibly connected to a daily commute, was centred close to the school they attended. During some runs of the k-means algorithm, the OpenStreetMaps API correctly identified the school by name. It is likely that this would have been achieved more reliably if MobileMiner recorded full GPS locations, as many commercial apps do. There were three other clusters of note, each in a UK city, with visits on two days each. Given that the user was attending a school, over 16 years of age, and the three cities hosted major universities, a reasonable assumption would be that they were attending open days, then interviews at the universities. This was further reinforced by the appearance of these universities in the logs of wireless networks, and confirmed when the user was interviewed at the second hack-day [5].

4.2 Predicting User Behaviour

The basic approach of examining the distribution of times of measurements within the spatial clusters yielded detailed and accurate information about users' behaviour. However, it would be attractive to predict behaviour as well as summarize it. The cluster identified as a school was occupied as late as 19:00, rather late for the UK school system. This may have been caused by an after-school activity regularly held on certain days of the week. Cluster occupancy was modeled using SciKitLearn's [27] implementation of Random Forests. A sequence of over 15400 cell tower IDs and their hour and day of occupancy for a single user were split into training and test data-sets in the ratio 4 to 1. Using the hour and day of the week as features allowed the cell tower ID to be predicted on only 20% accuracy. This made no attempt to use the physical locations of cell towers or deal with fluctuations in connections. The cell IDs were then replaced with their cluster-labels from the position and velocity based k-means clustering. Since only cells known to the OpenCellID database could be used, the data-set was reduced to a size of 5670 cell connections over 100 distinct days.

94% of cluster visits were to two clusters, with 62% and 32% of visits each. This time the same split between training and test data produced a Random

Forest classifier able to predict cluster occupancy for a given time and day with over 99.9% accuracy. Obviously, the data in this study is very heavily skewed, but the classifier could not achieve this by only correctly identifying the two most popular clusters. A dummy classifier that merely reproduced the class distribution of the training data could only achieve around 50% accuracy, naive Bayes scored 75% using the same features. Gatmir-Motahari *et al.* [28] conducted a study using a large amount of network operator data, achieving over 90% accuracy in place prediction having carefully considered factors including time of day, day of the week and conventional working hours. They concluded that their subjects may have led particularly regular lifestyles, others from more rural and less affluent areas may have been harder to predict. Song *et al.* [29] quote 93% as a more reasonable limit to predictability for cell tower traces. The random forest classifier may have over-fit significantly, and might not cope well with changes in routine. However, it does serve to demonstrate to subjects that their behaviour can be modeled with quite sparse and coarse data.

5 Conclusions

While small datasets from crowd-sourced collections of location and social media data cannot produce the insights of those from datasets where access is granted by service providers, they can give a good indication to participants of what insights are possible. MCL has shown how social media can identify groups with which users associate, and their social context. The knowledge gained about users by social media platforms is enhanced by their use on mobile devices when access to location data is granted. A simple two-step process of clustering cell tower trails followed by supervised learning of cluster occupancy over time is sufficient to demonstrate this; actively engaging participants in their attitudes to their data and privacy.

References

1. Levandoski, J., Sarwat, M., Eldawy, A., Mokbel, M.: LARS: a location-aware recommender system, pp. 450–461 (2012)
2. Wayback machine archive of the University of Minnesota website. <http://web.archive.org/web/20161202171606/http://www-users.cs.umn.edu/sarwat/foursquaredata/>
3. Young rewired state website. <https://yrs.io/>
4. Blanke, T., Greenway, G., Pybus, J., Cote, M.: Mining mobile youth cultures. In: 2014 IEEE International Conference on Big Data, pp. 14–17 (2014)
5. Pybus, J., Coté, M., Blanke, T.: Hacking the social life of big data. Big Data Soc. **2**(2) (2015). <https://doi.org/10.1177/2053951715616649>
6. Beer, D., Burrows, R.: Popular culture, digital archives and the new social life of data. Theor. Cult. Soc. **30**(4), 47–71 (2013)
7. Pybus, J.: Social networks and cultural workers. J. Cult. Econ. **6**(2), 137–152 (2013)
8. Conversocial website. <http://www.conversocial.com/>
9. Sproutsocial website. <http://sproutsocial.com/>

10. Twitonomy website. <http://www.twitonomy.com/>
11. Simplymeasured website. <http://simplymeasured.com/>
12. Preotiuc-Pietro, D., Samangooei, S., Cohn, T., Gibbins, N., Nirajan, M.: Trendminer: an architecture for real time analysis of social media text. In: 6th International AAAI Conference on Weblogs and Social Media (ICWSM 2012), June 2012
13. Big social data project website. <http://big-social-data.net/>
14. Bruns, A.: How long is a tweet? Mapping dynamic conversation networks on Twitter using Gawk and Gephi. *Inf. Commun. Soc.* **15**(9), 1323–1351 (2012)
15. Himmelboim, I., McCreery, S., Smith, M.: Birds of a feather tweet together: integrating network and content analyses to examine cross-ideology exposure on Twitter. *J. Comput. Med. Commun.* **18**(2), 40–60 (2013)
16. NodeXL website. <http://nodexl.codeplex.com/>
17. Van Dongen, S.: Graph clustering via a discrete uncoupling process. *SIAM J. Matrix Anal. Appl.* **30**(1), 121–141 (2008)
18. Opencellid website. <https://opencellid.org/>
19. OpenStreetMap Contributors: Planet dump (2017). <https://planet.osm.org>, <https://www.openstreetmap.org>
20. Isaacman, S., Becker, R., Cáceres, R., Kobourov, S., Martonosi, M., Rowland, J., Varshavsky, A.: Identifying important places in people's lives from cellular network data. In: International Conference on Pervasive Computing, pp. 133–151 (2011)
21. van der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy array: a structure for efficient numerical computation (2011)
22. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks (2009)
23. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.* **2008**(10), P10008 (2008)
24. Ingersoll, G.S., Morton, T.S., Farris, A.L.: Taming Text. Manning Publications, Shelter Island (2013)
25. Schaeffer, S.E.: Graph clustering. *Comput. Sci. Rev.* **1**(1), 27–64 (2007)
26. Brandes, U., Gaertler, M., Wagner, D.: Experiments on graph clustering algorithms. In: Di Battista, G., Zwick, U. (eds.) *ESA 2003. LNCS*, vol. 2832, pp. 568–579. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39658-1_52
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
28. Gandomi-Motahari, S., Zang, H., Reuther, P.: Time-clustering-based place prediction for wireless subscribers. *IEEE/ACM Trans. Netw.* **21**(5), 1436–1446 (2013)
29. Song, C., Qu, Z., Blumm, N., Barabási, A.L.: Limits of predictability in human mobility. *Science* **327**(5968), 1018–1021 (2010)

Churn Prediction Using Dynamic RFM-Augmented Node2vec

Sandra Mitrović^{1(✉)}, Bart Baesens^{1,2}, Wilfried Lemahieu¹,
and Jochen De Weerdt¹

¹ Department of Decision Sciences and Information Management,
KU Leuven, Leuven, Belgium
sandra.mitrovic@kuleuven.be

² School of Management, University of Southampton, Southampton, UK

Abstract. Current studies on churn prediction in telco apply network analytics to analyze and featurize call graphs. While the suggested approaches demonstrate a lot of creativity when it comes to deriving new features from the underlying networks, they also exhibit at least one of the following problems: they either do not account properly for dynamic aspects of call networks or they do not exploit the full potential of joint interaction and structural features and additionally, they usually address these in a non-systematic manner which involves hand-engineering of features. In this study, we propose a novel approach in which we address each of the identified issues. In a nutshell, first, we propose slicing a monthly call graph to capture dynamic changes in calling patterns. Second, we devise network designs which conjoin interaction and structural information. Third, we adapt and apply the node2vec method to learn node representations in a more automated way and to avoid the need for feature handcrafting.

Keywords: Dynamic networks · Graph featurization
Learning node representation · Churn prediction in telco

1 Introduction

Recent works on churn prediction in telco have been focusing on using network analytics to exploit Call Detail Records (CDR) [6, 10, 13, 16, 33]. However, extracting features from networked data is not a straightforward process and in case of very large graphs also comes at a huge computational expense. The reasons for this are varied. First, call networks are complex data structures which can be characterized both by structural features of the underlying graph topology and interaction features which, per customer, aggregate customer behavior from CDR data. Current studies address the extraction of these features in an ad-hoc manner using handcrafted features [6, 10, 16, 21, 22, 28, 33], thereby balancing between computational overload and predictive performance [34]. Computational overload is especially important in the case of structural features. On the other

hand, interaction features are usually handled by different operationalizations of the well-known RFM (**R**ecency, **F**requency, **M**onetary) model [5, 11].

An additional problem is that current studies exploiting network analytics for churn prediction in telco consider call graphs as static networks, only taking into account the network snapshot at the end of the observed period. However, call graphs are rather dynamic as they gradually evolve over time¹ [3]. While a majority of works on dynamic networks only considers the addition and removal of edges over time [19, 23, 29], the changes of topology in the CDR graphs reflect both the volatility of graph nodes and the links between them. For example, in a mobile network, a customer C_i might start calling another customer C_j (adding an edge), or customer C_k might stop calling customer C_l (removing an edge); likewise, a new customer C_s could join the network (adding a node) or an existing customer C_t can decide to churn (removing a node). Therefore, detecting the changes in the calling patterns proves essential for predicting churn.

In this work, we aim at performing holistic featurization of call graphs by incorporating both interaction and structural information while also capturing the dynamic aspect of the network. We perform this by devising a novel approach, as follows. First, we formulate different RFM operationalizations based on the granularity of interaction information. Second, to incorporate interaction and structural information, we construct RFM-augmented CDR networks. Third, to account for the dynamic aspect, we consider four one-week dynamic networks instead of a single monthly network. Fourth, to avoid handcrafting of features, we adapt node2vec [8] and make it scalable to our setting. Node2vec is a fairly recent but already well-known node representation learning approach.

Our research questions are positioned along three main directions. First, our main research question is whether dynamic node representations approximated by representations obtained from four one-week networks can lead to better predictive performance as compared to a static node representation generated from a one month network. Second, we would like to investigate whether our RFM-augmented networks allow for higher predictive performance than standard RFM variables. Third, we are also interested in knowing whether the granularity of interaction information influences the predictive power.

Our results show that taking into account the dynamic aspect improves predictive performance. Additionally, we demonstrate that designed RFM-augmented extensions of the original graphs improve AUC and lift scores. To the best of our knowledge, this paper is the first in using dynamic node representations in CDR graphs for churn prediction and the first in applying the RFM framework together with unsupervised and dynamic learning of node representations.

The rest of the paper is organized as follows. In Sect. 2, we provide a short overview of literature concerning our topics of interest. In Sects. 3 and 4, we explain our method and experimental setup, respectively. In Sect. 5 we present the results of our experiments and we provide a conclusion and ideas for future research in Sect. 6.

¹ The terms dynamic, temporal, (time-)evolving, time-varying (networks) are also used interchangeably in the literature.

2 Related Work

The related literature overview is focused on the areas which are of interest for this work: dynamic networks, interaction features, structural features and representation learning in graphs. Likewise, whenever possible, the focus is put on churn prediction in telecommunications.

Exploiting Dynamic Networks. A number of studies have already demonstrated that, in the context of analyzing customer behavior in telco, taking into account the dynamic/temporal nature of customer interactions improves predictive performance as compared to static approaches [4, 7, 9, 15]. Nevertheless, these works mostly apply time series techniques [4, 17, 24] and only to a limited extent (if at all) consider changes in network topology. On the other side, applications of dynamic networks to CDR graphs are very rare in the literature. In [9], dynamic networks were applied on call graphs in the context of fraud detection. In their approach, daily increments (used as timestamps) are combined using exponential smoothing to penalize historical data and the resulting graph is additionally pruned, using tuned parameter values.

One of the standard approaches in handling temporal networks is to consider several static snapshots of dynamic graphs, i.e. a sequence of static graphs, and then to combine information which has been derived from these static graphs [3, 19, 23]. In our study, we follow this direction, which has recently been used in a link prediction context as well [29]. However, to the best of our knowledge, this is the first work which explicitly exploits dynamic networks (and dynamic node representations) for churn prediction in telco.

Capturing Interactions with the RFM Model. The RFM model is a prevailing method used in the literature to characterize customer interactions and customer behavior in general. It is a simple, yet powerful method which enables quantifying customer interactions in terms of recency, frequency and monetary dimensions. The current literature related to churn prediction in telco devises many different RFM operationalizations, ranging from summary, coarse-grained to more fine-grained features. Examples of summary RFM variables are total call frequency, total call volume (seconds) [6], seconds of use, frequency of use and frequency of SMS in [13]. Examples of fine-grained RFM features include features obtained by slicing along the time dimension, e.g. minutes of usage by days of the week (working days and weekend) in [22, 25]; along the direction dimension, e.g. total incoming/outgoing call duration (seconds) as defined in [6, 21]; along the types of counterparties both at destination e.g. in-net call duration in [12] and churner² level e.g. total interaction frequency with churners in [6, 16], or a combination of these, e.g. number of incoming, outgoing calls to/from a different operator's network in [6, 21, 22]. Additionally, RFM features are engineered both by absolute values and percentages, e.g. the call frequency percentage (with respect to total) to/from a different operator's network in [6].

² A chunner is an individual who has stopped using mobile operator services.

Extracting Structural Features. Although to a lesser extent than interaction features, structural features are also used in many previous works on churn prediction in telco in order to capture the topology of the underlying graph. Almost every study exploits 1st-order in/out degree. Even though some studies use more complex structural features, such as 2nd and 3rd-order degree in [16]; 2nd-order degree and a clustering coefficient in [33]; PageRank in [10]; degree centrality, closeness centrality, eccentricity centrality, clustering coefficient, Shapley value, degree and proximity prestige in [30], the main issue with structural features is their computational complexity due to which, despite recognizing their predictive power, many studies do not take them into account. This is for instance the case with closeness and betweenness centrality in [34]. The potential of structural features has definitely not been fully exploited in the literature. Unlike the previously mentioned works, in our study, we do not perform hand-engineering neither do we limit the number of structural features. Structural features are, instead, embedded in the learned node representations.

Combining Interaction and Structural Information. Several studies on churn prediction in telecommunications have already used both interaction and structural information [6, 10, 16, 21, 22, 33]. However, these studies mostly exploit either only degree measures [6, 16, 21, 22] or a slightly extended, but still limited number of structural features [10, 33]. Some studies use these features implicitly (e.g. the number of calls that neighbors of one node make to the neighbors of another node i.e. frequency based on 2nd-order neighborhood in [28]). In [27], customer behavior is considered to be independent from the call graph structure but, nonetheless, degree centrality is used to normalize the values of the interaction features. To the best of our knowledge, no telco churn prediction related study carried out an inquiry to determine which of these two classes of features provides better predictive scores. On the other hand, RFM variables were found to have more importance than structural features in the task of classifying edges in a telco graph [28] and churn prediction in banking [2]. This provides a good motivation for enriching graph topologies with RFM variables.

Representation Learning on Graphs. Representation learning is a fastly growing field of research aiming not just to automate the feature engineering process, but also to acquire task-independent and high performing feature representations. It has many applications in various domains such as natural language processing, speech recognition and image classification. In fact, the most relevant recent work with respect to our study is presented in [8, 26], and introduces representation learning on graphs, based on an equivalent model first developed in a natural language processing (NLP) context [20]. Known applications of node representations include multi-label classification in social networks (Facebook [8], Flickr [26, 31], YouTube [26, 31], blogger networks derived from the BlogCatalog website [8, 26]), Wikipedia words co-occurrence networks [8, 31] and citation networks (DBLP) [31], as well as link prediction on the Facebook graph, Protein-Protein Interaction graphs and a collaboration graph from ArXiv [8]. However,

to the best of our knowledge, representation learning has not been used before nor in the context of churn prediction in telco neither in a combination with RFM variables. In addition, this paper uses dynamic node representations in CDR graphs for telco churn prediction.

3 Dynamic and Scalable RFM-Augmented Node Representation Learning

In this section, we introduce our approach, which is based on several important concepts. First and foremost, we aim at exploiting the dynamic nature of CDR graphs for which we split the original monthly CDR data into weekly subsets, thus inducing smaller graphs. Second, we conjoin both structural and interaction information by devising different RFM operationalizations and network designs. Third, we circumvent the usual practice of handcrafting network features by applying a scalable version of an existing node representation learning method. These concepts are detailed in the following sections while the overview of our approach can be seen in Fig. 1.

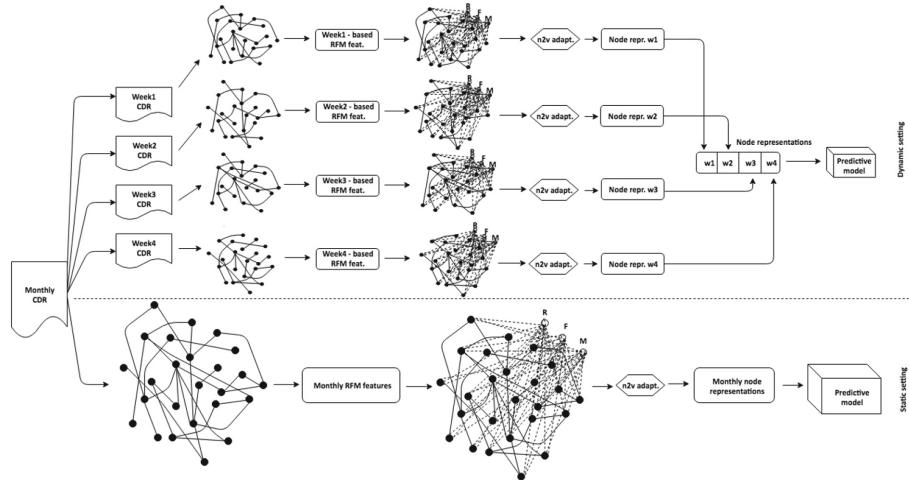


Fig. 1. At the top of the figure, our dynamic and scalable RFM-augmented node representation learning approach is shown. First, monthly CDR data is split in weekly partitions and used to generate weekly networks. From these, RFM information is derived and used to construct RFM-augmented weekly networks. Next, our adaptation of node2vec is applied and the obtained node representations are concatenated into a 4×128 -long feature vector, which is then provided as input to a predictive model. At the bottom, the static equivalent is depicted.

3.1 Exploiting Dynamicity in CDR Graphs

The comprehensive formalization of dynamic graphs is still an on-going work, even though their application is in expansion in different research areas. Existing literature formally defines a dynamic graph either as a triplet $G = (V, E, T)$ [19], a quadruplet $G = (V, E, T, \Delta T)$ [23] or even quintuplet $G = (V, E, T, \sigma, \Delta T)$ [3], where V represents the set of nodes, E the set of edges, T the set of moments in time when the edges appear, σ the edge presence function indicating if an edge is present at a particular moment, ΔT the time interval during which the edge (relation/interaction) exists.

In order to capture the dynamic nature of relationships between customers, in our approach, we split a monthly CDR graph $\mathcal{G} = (V, E)$ into four temporal graphs $G_i = (V_i, E_i, w_i)$, $i \in \{1, \dots, 4\}$, where w_i represents the i^{th} week³, V_i represents all customers having calls (i.e. appearing in the CDR) during the week i , E_i represents interactions between customers during the week i . Naturally, $V = \bigcup_{i \in \{1..4\}} V_i$, $E = \bigcup_{i \in \{1..4\}} E_i$ and $T = \bigcup_{i \in \{1..4\}} w_i$. However, unlike many dynamic networks, in our case it can occur that $V_i \neq V_j$ for $i \neq j$, due to the fact that existing customers do not have to have a call in every week of the month and, new customers can join the operator network. Additionally, notice that instead of a more detailed time granularity (e.g. day-level in [9]), we opt for week-based intervals. Such an approach leads to less sparsity in the obtained networks, which is essential for our node representation algorithm. Moreover, as we address interaction data through RFM features, we do not include duration of interactions (ΔT) in the definition of temporal graphs.

Once the dynamic networks are featurized in a way which will be explained in the following subsections, the resulting feature-set f_c for customer c is obtained by concatenating the corresponding weekly features, i.e. $f_c = \oplus_{i \in \{1, \dots, 4\}} f_c^{w_i}$, where \oplus represents the concatenation operator and $f_c^{w_i}$ is the representation for customer c learned on a dynamic network corresponding to week w_i . This setting has been adopted in previous studies as well, e.g. in [29].

3.2 Operationalizing RFM Network Features

We use the following definitions of RFM variables to quantify customer interaction behavior on a per-customer and per-observed period (week) basis:

- Recency: The number of days between the end of the observed period and the customer’s last call (within the same period).
- Frequency: The number of calls of a customer during the observed period.
- Monetary: The duration (in seconds) of customer calls during the observed period.

Although numerous RFM operationalizations have been proposed in recent literature (as already mentioned in Sect. 2), we consider the following three variants, which despite their simplicity do convey relevant interaction information:

³ The fourth week is considered to begin at 22nd day and lasts till the end of the month, hence longer than three previous weeks.

- Summary-RFM (denoted by RFM_s): total R/F/M per customer per observed period, i.e. overall recency (R), total number of calls (F) and total duration (M).
- Detailed-RFM (denoted by RFM_d): each of the R/F/M variables is sliced based on the direction and destination dimension into three subcategories: outgoing towards home network, outgoing towards other networks and incoming (denoted as $R_{out,h}, R_{out,o}, R_{in}$; similarly for M and F), inspired by the approaches in [6, 12, 21, 22].
- Churn-RFM (denoted by RFM_{ch}): R/F/M variables calculated with respect to customers who churned (denoted as R_{ch}, F_{ch}, M_{ch}). Hence, in this case, M_{ch} , for example, represents the total duration of calls to/from churers. A similar characterization of RFM has been done in [6, 21, 33].

3.3 Constructing RFM-Augmented Networks

Our dynamic networks are constructed in a three-step procedure. First, we construct weekly graphs using a conventional way of representing customers as nodes and adding edges only between the nodes whose corresponding customers were involved in a call during the observed weeks. Second, we calculate different RFM variants as explained in the previous paragraph. Third, we incorporate the obtained RFM features to enrich the original graphs and obtain new RFM-augmented network designs.

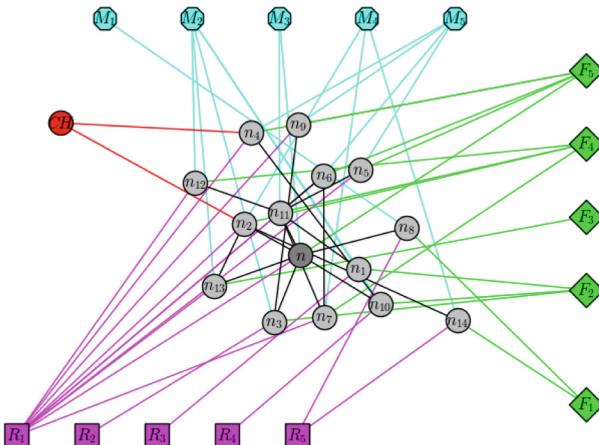


Fig. 2. An extract from the AG_{s+ch} network, depicting a 2^{nd} -level neighborhood of a node n , with artificial nodes magenta (square), green (diamond) and blue (octagon) representing R(ecency), F(requency) and M(onetary), respectively. R_i corresponds to i^{th} quintile with respect to R(ecency), similarly for F_i and M_i . The artificial node corresponding to churers is denoted with CH in red. (Color figure online)

The idea of RFM-augmented networks is to enrich the original topology by adding RFM information in the form of additional artificial nodes (Fig. 2). To

devise the construction of these artificial nodes, we follow the idea frequently used in the literature related to customer segmentation and customer lifetime value modeling, where customers are usually segmented partitioning each of their R/F/M variables in five equally-sized groups (corresponding to very high, high, medium, low, very low) [5, 11, 18]. Inspired by this approach, we devise four different networks, as follows:

- Augmented Graph with Summary-RFM Artificial Nodes (AG_s)

In this graph, first, one new (artificial) node is introduced for each quintile, obtaining thus 15 artificial nodes R_i, F_i, M_i , where $i \in \{1, 2, \dots, 5\}$ ⁴. Next, corresponding edges between original nodes and artificial nodes are added, connecting each node to exactly one node from the set of artificial R nodes, exactly one node from the set of artificial F nodes and exactly one node from the set of artificial M nodes, based on the appropriate R, F and M quintiles. In this way, the newly obtained graph has at most 15 new nodes, but exactly $3 * |V|$ more edges than the original one (with $|V|$ being the number of nodes in the original graph).

- Augmented Graph with one Churn Node and Summary-RFM Artificial Nodes (AG_{s+ch})

This graph is exactly the same as AG_s , except that one additional artificial node representing churn is added, to which then all the churners are connected. By adding this node, we try to compensate for the information contained in churn-RFM related features, which we do not exploit in this case as we try to keep the number of nodes and edges in the resulting graph not too large.

- Augmented Graph with Detailed-RFM Artificial Nodes (AG_d)

In this graph, we repeat the steps from AG_s , except that unlike AG_s where R/F/M correspond to Summary-RFM, here we start from Detailed-RFM information, calculating each of R, F, M on a more fine-grained level using direction and destination dimensions to obtain three categories: incoming, outgoing towards the home network and outgoing towards other networks. Then each of these three categories is binned into quintiles, thus leading to 45 artificial nodes which are used to enrich the original graph. As with AG_s , each node is connected to the appropriate artificial nodes. The number of additional edges, compared to the original graph, increases with nine times the original number of nodes (since each node becomes connected to three more R artificial nodes - for incoming, home outgoing and other outgoing, and similarly 3 more per F and per M).

- Augmented Graph with one Churn Node and Detailed-RFM Artificial Nodes (AG_{d+ch})

Similarly to AG_{s+ch} , this graph is exactly the same as AG_d , except for adding one additional artificial node representing churners, to which then all the churners are connected.

⁴ Exceptionally, due to the skewed distribution of R/F/M values, one can end up having less than five quintiles per R/F/M.

All constructed networks are considered to be undirected. This decision is motivated by the fact that our call graph is sparse, and hence, retaining directed graphs would make random walks get stuck at sink nodes. In addition, the preference to undirected call graphs was also observed in previous works, where only handcrafted features were utilized (without considering random walks) [14]. Nevertheless, an attempt to consider directed networks was made as well, but the obtained performance was sub-par.

Additionally, we consider augmented graphs unweighted, for two different reasons. First, due to a very different nature between the type of nodes (“real” vs. artificial), it is not easy to determine the corresponding weights as we would not like to bias the process of walk generation neither towards “real” nor towards the artificially added nodes. Second, it is not very clear how to determine the weights between artificial nodes since the current RFM-related literature usually quantifies the relation between RFM variables in an ad-hoc and domain/dataset-dependent manner. To avoid this, some previous studies considered them with equal priority [5, 18] and we decided to follow the same approach.

3.4 Scalable Node2vec-Based Representation Learning

Our node representation learning algorithm is a modification of the node2vec implementation provided by [8]. Node2vec (as well as some other previous works [26, 31]) draws a parallel between the context of a word in a document and a neighborhood of a node in the graph, and due to this similarity, learns node representations using the SkipGram model [20]. The SkipGram model aims to maximize the probability of collocating the words from the same context. More precisely, it brings the representations of the words from the same context closer than those of the words found in different contexts. More precisely, if we denote a set of words in a vocabulary by V (or analogically, a set of nodes in the graph), we are learning a function $f : V \rightarrow \mathbb{R}^d$ such that

$$\max \sum_{v \in V} \log Pr(C_v | f(v))$$

where C_v is a context of word (node) v . This objective function is further simplified using the independence assumption, while the conditional likelihood is modeled with a softmax function in which a computationally expensive normalization factor is approximated using negative sampling (more details can be found in [8, 20]).

However, the generation of a node context, C_v , in a graph setting is not as straightforward as a word context in the NLP setting. Most of the studies dealing with this issue opt for generating node contexts (neighborhoods) using fixed-length random walks [8, 26]. In order to allow for flexibility and finding the best balance between breadth-first and depth-first sampling strategies during random walks, a return parameter p and a in-out parameter q were introduced

in the node2vec approach. However, this flexibility comes with additional computational burden as it not only requires precomputing transition probabilities to allow for efficient sampling afterwards, but also necessitates constructing two alias tables, one for nodes and one for edges (the alias method is used for sampling). Therefore, in the provided implementation (from [8]), the node2vec approach becomes computationally unfeasible in the case of very large graphs (with e.g. 40M of edges, as in our case). Hence, in our approach, we simplify the random walk construction defining the probability of moving from a certain node to its neighboring node as the normalized weight of the corresponding adjacent edge. This modification allows for faster walk generation and significantly decreases computational time since it eliminates the necessity of precomputing the alias table for the edges. Therefore, our method is comparable to the DeepWalk [26] and LINE [31] approaches. However, the difference between our approach and DeepWalk consists in the way of approximating the normalization factor in the softmax probabilities. In our approach, as inherited from node2vec, negative sampling is used, in contrast to the less efficient hierarchical softmax, used in DeepWalk. The differences between our method and LINE are more substantial. Namely, in our approach, the context (walk) of full (predefined) length is generated in one pass after which its representation is learnt by the SkipGram model. In contrast to this, in LINE, each context is generated from two parts, which are constructed independently from the first and second-order neighborhood and additionally, optimized separately using two different objective functions.

3.5 Churn Definition

Given that the provided dataset does not contain churn labels, we opt for defining churn based on the absence of a customer’s activity, which can be observed from CDRs, in line with previous works [1, 16]. However, previous studies usually use the complete data from one month to predict which customers will churn immediately during the next month. These scenarios are not applicable in real business cases as they leave no time for devising appropriate marketing retention campaigns. Therefore, in this work, we apply a different setting using four months of data, denoted as $M - 1$, M , $M + 1$, $M + 2$, in the following way (see also Fig. 3):

- Customers appearing in the CDR of month M are considered to constitute a customer base
- For each customer belonging to this customer base, we are predicting the probability of churn during the first three weeks of month $M + 2$
- In the static setting, each customer from month $M - 1$ is marked as churner during month M , if:
 - The customer appears in the CDR during the month M and has not ported out (decided to switch to another provider while retaining the same phone number) during the month M
 - The customer does not appear in the CDR during the first three weeks of month $M + 1$ or has been ported out during the same period

- In the dynamic setting, a customer from month $M - 1$ is marked as churner during week w_i of month M , if:
 - The customer satisfies the same criteria as above for the static setting, and
 - The customer had its last call in month M during week w_i

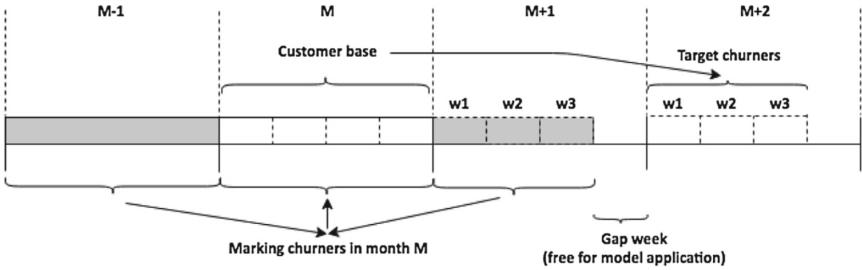


Fig. 3. Graphical illustration of the four-month CDR information usage.

The last criterion in the dynamic setting ensures that the correct churn date is taken into account in the case when a customer had activities in several weeks of the same month.

4 Experimental Setup

In this section, we provide a brief description of our datasets, experiments, predictive model, different parameterizations and evaluation method used.

4.1 Data

We perform our experiments on one prepaid and one postpaid dataset, each of which consists of four consecutive months of CDR data. The only usage type available in CDRs are calls (no SMS/MMS/GPRS usage), for which we are provided with (anonymized) information about caller, callee, as well as the date and the (real) duration of calls. For postpaid customers, we also have information about whether and when the customer has ported-out, but this is the case for only a dozen of customers on a monthly level.

Applying the before mentioned churn definition, we end up with around 12.2% of churners for the prepaid and 7.8% of churners for the postpaid dataset (more details on datasets statistics can be found in Table 1).

Table 1. Statistics of the datasets used, both for monthly and weekly level.

Measure	Prepaid					Postpaid				
	Month	Week 1	Week 2	Week 3	Week 4	Month	Week 1	Week 2	Week 3	Week 4
Number of nodes	4303541	2251195	2213089	2197238	2731485	4799149	2929347	2889360	2894179	3414656
Number of edges	5936423	2116619	2053684	2031695	2817743	9246134	3488464	3404493	3411408	4556203
Average degree	2.75886	1.88044	1.85594	1.84932	2.06316	3.85324	2.38173	2.35657	2.35743	2.66862
Average clustering coefficient	0.05749	0.03121	0.03029	0.02958	0.03950	0.06939	0.04450	0.04405	0.04295	0.05370
Number of connected components	138509	314793	322511	320098	266883	27392	143390	149481	148274	97970
Size of maximal clique	7	6	6	6	6	7	6	5	6	6
Degree assortativity coefficient	-0.00110	-0.00145	-0.00144	-0.00141	-0.00138	-0.02290	-0.01951	-0.01937	-0.01954	-0.02073

4.2 Experiments

In this study, we aim at examining and comparing the predictive power of features along three main directions. First, we would like to investigate whether dynamically derived features perform better than static ones. Second, we would like to analyze which feature origination method (classical RFM or augmented networks) works best. Third, we are also interested in knowing whether the granularity of interaction information (summary, summary+churn information, detailed, detailed+churn) influences the predictive performance. Therefore, we consider a full factorial design consisting of three factors:

- F1: Temporal factor with two levels: static and dynamic;
- F2: Feature origination factor, with two levels: classical RFM (*RFM*) and augmented networks with adapted node2vec (*AG*);
- F3: Granularity of interaction information, containing four levels: summary (*s*), summary+churn information (*s + ch*), detailed (*d*), detailed+churn (*d + ch*).

In the end, we come up with $2 * 2 * 4 = 16$ different combinations per dataset.

Although we focus on comparisons along the mentioned factors, we also perform cross-comparison of different combinations.

It is important to mention that two sets of parameters are used. The first one is related to our adapted version of the original node2vec method and consists of the number of walks n and walk length l , instantiated as: $n = 10$ and $l = 30$ (experimentally evaluated). The second one refers to the underlying SkipGram model and consists of three parameters: the number of iterations i , the context window size s , and the number of dimensions in the resulting representation d . For these, we set $i = 5$, $s = 10$ and $d = 128$.

4.3 Model and Evaluation Methods

Our predictive models are generated using logistic regression with l_2 -regularization. Logistic regression is a well-known classification technique applied in numerous studies related to telco churn prediction [1, 10, 14, 21, 25, 33]. Additionally, previous studies performing classification based on learned node representations also used logistic regression [8, 26]. The advantage of logistic regression

is that it is more easily interpretable, while it performs on-par with more complex techniques [25, 32]. We use 10-fold cross validation to tune the regularization hyper-parameter.

The AUC and lift scores (at 0.5%) are used for evaluation which is performed in an out-of-sample fashion.

5 Experimental Results

In this section, we present the results of 10-fold cross validation (with different folds than in the case of hyperparameter tuning) as can be seen in Tables 2 and 3 for prepaid and postpaid, respectively.

Regarding the temporal factor (F1) where we consider static and dynamic levels, except for the lift measure in case of the prepaid AG_s network, features taking into account the dynamic aspect always perform better in terms of both AUC and lift and both for prepaid and postpaid, as can be seen in Tables 2 and 3.

Table 2. Comparison in terms of AUC and lift (at 0.5%) between different methods for the **prepaid** dataset. The results are averaged across 10 folds (different from folds used for hyperparameter tuning). Horizontally, a comparison between the different feature origination methods (RFM, augmented networks) for the same interaction granularity is provided. Vertically, a comparison between the different interaction granularities (summary/summary+churn/detailed/detailed+ch) for the same RFM and/or network type can be performed. Finally, a static vs. dynamic comparison for each different setting can be performed by simply looking at adjacent columns. The best score per row is marked in bold. The best overall score is marked in bold and underlined.

RFM	Static		Dynamic		Augmented network	Static		Dynamic	
	AUC	Lift	AUC	Lift		AUC	Lift	AUC	Lift
RFM_s	0.671	<u>1.788</u>	0.680	2.025	AG_s	0.680	2.061	0.694	2.013
RFM_{s+ch}	0.671	1.789	0.689	2.014	AG_{s+ch}	0.680	1.976	0.705	<u>2.331</u>
RFM_d	0.683	1.857	0.692	2.063	AG_d	0.678	1.898	0.693	2.019
RFM_{d+ch}	0.682	1.856	0.695	2.040	AG_{d+ch}	0.680	1.967	0.702	2.316

With respect to feature origination (F2), where we distinguish between classical RFM (RFM_*) and augmented networks (AG_*), with $* \in \{s, s+ch, d, d+ch\}$, we can see that both in terms of AUC and lift, features obtained from augmented networks score better than the classical RFM features (for prepaid, the highest AUC/lift for RFM_* is 0.695/2.063, which is lower than the AUC/lift for AG_* being 0.705/2.331; similarly for postpaid, AUC/lift for RFM_* is 0.767/3.885 which is lower than 0.769/3.928 for AG_*).

Finally, when analyzing the granularity of interaction information (F3) whereby we consider four levels: summary (s), summary+churn information

Table 3. Comparison in terms of AUC and lift (at 0.5%) between different methods for the **postpaid** dataset. The results are averaged across 10 folds (different from folds used for hyperparameter tuning). Horizontally, a comparison between the different feature origination methods (RFM, augmented networks) for the same interaction granularity is provided. Vertically, a comparison between the different interaction granularities (summary/summary+churn/detailed/detailed+ch) for the same RFM and/or network type can be performed. Finally, a static vs. dynamic comparison for each different setting can be performed by simply looking at adjacent columns. The best score per row is marked in bold. The best overall score is marked in bold and underlined.

RFM	Static		Dynamic		Augmented network	Static		Dynamic	
	AUC	Lift	AUC	Lift		AUC	Lift	AUC	Lift
RFM_s	0.741	<u>3.367</u>	0.743	3.403	AG_s	0.759	3.602	0.768	<u>3.919</u>
RFM_{s+ch}	0.741	3.369	0.758	3.858	AG_{s+ch}	0.760	3.553	<u>0.769</u>	<u>3.928</u>
RFM_d	0.750	3.750	0.757	3.874	AG_d	0.754	3.716	0.764	<u>3.908</u>
RFM_{d+ch}	0.750	3.751	0.767	3.885	AG_{d+ch}	0.755	3.720	0.764	<u>3.901</u>

($s + ch$), detailed (d), detailed+churn ($d + ch$), we notice that for both datasets, features derived based on summary+churn information ($s + ch$) perform best in terms of AUC and lift. However, the rankings of the levels differ among the two datasets. More precisely, for prepaid the second best level both in terms of AUC and lift is $d + ch$, then in terms of AUC the third best is s while the worst is d , while in terms of lift the opposite holds. On the other hand, for postpaid, the second best level is s , both in terms of AUC and lift, while the third best in terms of AUC is $d + ch$ and in terms of lift is d .

6 Conclusion and Future Work

In this work, we devise a novel approach which allows for a holistic featurization of call graphs by incorporating the dynamic aspect with both interaction and structural information. We construct dynamic RFM-augmented CDR networks based on four different RFM operationalizations and circumvent traditional handcrafting of features by adapting the node2vec approach. The obtained results demonstrate that taking into account the dynamic aspect and RFM-augmented extensions of the original graphs improves predictive performance in terms of AUC and lift scores. This paper contributes to the literature in being the first to develop a model for dynamic node representations in CDR graphs to predict churn in telco. More specifically, the proposed approach builds node representations based on dynamic, RFM-augmented networks obtained from CDR data.

Constructing RFM-augmented networks and corresponding node representations on a weekly level is quite time consuming. On the other hand, it can be argued that summarizing customer behavior on a weekly level hinders the full

potential of CDR graph dynamics. Therefore, for future work, we would like to investigate how different time granularities influence the predictive score. Additionally, due to the relatively short time frame (four weeks), in this work we considered all dynamic networks as equally relevant. Nevertheless, it would be interesting to see if prioritizing more recent dynamic networks leads to performance improvement. Moreover, capturing call dynamics in a more sophisticated manner (e.g. the ordering of calls, their inter-event time distribution) shall be considered as well.

References

1. Backiel, A., Baesens, B., Claeskens, G.: Mining telecommunication networks to enhance customer lifetime predictions. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014. LNCS (LNAI), vol. 8468, pp. 15–26. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07176-3_2
2. Benoit, D.F., Van den Poel, D.: Improving customer retention in financial services using kinship network information. *Expert Syst. Appl.* **39**(13), 11435–11442 (2012)
3. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distrib. Syst.* **27**(5), 387–408 (2012)
4. Chen, Z.Y., Fan, Z.P., Sun, M.: A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioral data. *Eur. J. Oper. Res.* **223**(2), 461–472 (2012)
5. Cheng, C.H., Chen, Y.S.: Classifying the segmentation of customer value via RFM model and RS theory. *Expert Syst. Appl.* **36**(3), 4176–4184 (2009)
6. Dasgupta, K., Singh, R., Viswanathan, B., Chakraborty, D., Mukherjea, S., Nanavati, A.A., Joshi, A.: Social ties and their relevance to churn in mobile telecom networks. In: Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, pp. 668–677. ACM (2008)
7. Eichinger, F., Nauck, D.D., Klawonn, F.: Sequence mining for customer behaviour predictions in telecommunications. In: Proceedings of the Workshop on Practical Data Mining at ECML/PKDD, pp. 3–10 (2006)
8. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. ACM (2016)
9. Hill, S., Agarwal, D.K., Bell, R., Volinsky, C.: Building an effective representation for dynamic networks. *J. Comput. Graph. Stat.* **15**(3), 584–608 (2006)
10. Huang, Y., Zhu, F., Yuan, M., Deng, K., Li, Y., Ni, B., Dai, W., Yang, Q., Zeng, J.: Telco churn prediction with big data. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 607–618. ACM (2015)
11. Hughes, A.M.: Strategic database marketing: the masterplan for starting and managing a profitable. Customer-based Marketing Program, Irwin Professional (1994)
12. Hung, S.Y., Yen, D.C., Wang, H.Y.: Applying data mining to telecom churn management. *Expert Syst. Appl.* **31**(3), 515–524 (2006)
13. Keramati, A., Jafari-Marandi, R., Aliannejadi, M., Ahmadian, I., Mozaffari, M., Abbasi, U.: Improved churn prediction in telecommunication industry using data mining techniques. *Appl. Soft Comput.* **24**, 994–1012 (2014)

14. Kim, K., Jun, C.H., Lee, J.: Improved churn prediction in telecommunication industry by analyzing a large network. *Expert Syst. Appl.* **41**(15), 6575–6584 (2014)
15. Klepac, G.: Discovering behavioural patterns within customer population by using temporal data subsets. In: *Handbook of Research on Advanced Hybrid Intelligent Techniques and Applications*, pp. 216–252. IGI Global (2016)
16. Kusuma, P.D., Radosavljevik, D., Takes, F.W., van der Putten, P.: Combining customer attribute and social network mining for prepaid mobile churn prediction. In: *Proceedings of the 23rd Annual Belgian Dutch Conference on Machine Learning (BENELEARN)*, pp. 50–58 (2013)
17. Lee, Y.H., Wei, C.P., Cheng, T.H., Yang, C.T.: Nearest-neighbor-based approach to time-series classification. *Decis. Support Syst.* **53**(1), 207–217 (2012)
18. McCarty, J.A., Hastak, M.: Segmentation approaches in data-mining: a comparison of RFM, chaid, and logistic regression. *J. Bus. Res.* **60**(6), 656–662 (2007)
19. Michail, O.: An introduction to temporal graphs: an algorithmic perspective. *Internet Math.* **12**(4), 239–280 (2016)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems*, pp. 3111–3119 (2013)
21. Modani, N., Dey, K., Gupta, R., Godbole, S.: CDR analysis based telco churn prediction and customer behavior insights: a case study. In: Lin, X., Manolopoulos, Y., Srivastava, D., Huang, G. (eds.) *WISE 2013. LNCS*, vol. 8181, pp. 256–269. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41154-0_19
22. Motahari, S., Jung, T., Zang, H., Janakiraman, K., Li, X.Y., Hoo, K.S.: Predicting the influencers on wireless subscriber churn. In: *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 3402–3407. IEEE (2014)
23. Nicosia, V., Tang, J., Mascolo, C., Musolesi, M., Russo, G., Latora, V.: Graph metrics for temporal networks. In: Holme, P., Saramäki, J. (eds.) *Temporal Networks*, pp. 15–40. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36461-7_2
24. Orsenigo, C., Vercellis, C.: Combining discrete svm and fixed cardinality warping distances for multivariate time series classification. *Pattern Recogn.* **43**(11), 3787–3794 (2010)
25. Owczarczuk, M.: Churn models for prepaid customers in the cellular telecommunication industry using large data marts. *Expert Syst. Appl.* **37**(6), 4710–4712 (2010)
26. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710. ACM (2014)
27. Phadke, C., Uzunalioglu, H., Mendiratta, V.B., Kushnir, D., Doran, D.: Prediction of subscriber churn using social network analysis. *Bell Labs Tech. J.* **17**(4), 63–75 (2013)
28. Raeder, T., Lizardo, O., Hachen, D., Chawla, N.V.: Predictors of short-term decay of cell phone contacts in a large scale communication network. *Soc. Netw.* **33**(4), 245–257 (2011)
29. Rahman, M., Hasan, M.A.: Link prediction in dynamic networks using graphlet. In: Frasconi, P., Landwehr, N., Manco, G., Vreeken, J. (eds.) *ECML PKDD 2016. LNCS (LNAI)*, vol. 9851, pp. 394–409. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46128-1_25

30. Saravanan, M., Vijay Raajaa, G.S.: A graph-based churn prediction model for mobile telecom networks. In: Zhou, S., Zhang, S., Karypis, G. (eds.) ADMA 2012. LNCS (LNAI), vol. 7713, pp. 367–382. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35527-1_31
31. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077. ACM (2015)
32. Verbeke, W., Dejaeger, K., Martens, D., Baesens, B.: Customer churn prediction: does technique matter? (2010)
33. Zhang, X., Zhu, J., Xu, S., Wan, Y.: Predicting customer churn through interpersonal influence. *Knowl.-Based Syst.* **28**, 97–104 (2012)
34. Zhu, T., Wang, B., Wu, B., Zhu, C.: Role defining using behavior-based clustering in telecommunication network. *Expert Syst. Appl.* **38**(4), 3902–3908 (2011)

Multi-scale Community Detection in Temporal Networks Using Spectral Graph Wavelets

Zhana Kuncheva^{1,2(✉)} and Giovanni Montana^{2,3}

¹ Clinical Development and Mathematics, C4X Discovery Ltd., London M1 3LD, UK
zhana.kuncheva@c4xdiscovery.com

² Department of Mathematics, Imperial College London, London SW7 2AZ, UK

³ Department of Biomedical Engineering,
King's College London, London SE1 7EH, UK
giovanni.montana@kcl.ac.uk

Abstract. Spectral graph wavelets introduce a notion of scale in networks, and are thus used to obtain a local view of the network from each node. By carefully constructing a wavelet filter function for these wavelets, a multi-scale community detection method for monoplex networks has already been developed. This construction takes advantage of the partitioning properties of the network Laplacian. In this paper we elaborate on a novel method which uses spectral graph wavelets to detect multi-scale communities in temporal networks. To do this we extend the definition of spectral graph wavelets to temporal networks by adopting a multilayer framework. We use arguments from Perturbation Theory to investigate the spectral properties of the supra-Laplacian matrix for clustering purposes in temporal networks. Using these properties, we construct a new wavelet filter function, which attenuates the influence of uninformative eigenvalues and centres the filter around eigenvalues which contain information on the coarsest description of prevalent community structures over time. We use the spectral graph wavelets as feature vectors in a connectivity-constrained clustering procedure to detect multi-scale communities at different scales, and refer to this method as Temporal Multi-Scale Community Detection (TMSCD). We validate the performance of TMSCD and a competing methodology on various benchmarks. The advantage of TMSCD is the automated selection of relevant scales at which communities should be sought.

Keywords: Temporal network · Multilayer network
Multi-scale community · Spectral graph wavelets

1 Introduction

Networks are used to model complex relationships in a wide range of real-life applications throughout the social, biological, physical, information technology and engineering sciences. Due to limitations in data collection and storage, mainly static (monoplex) networks have been studied. However, many real-world

systems have relationships between entities that evolve over time [15]. Technological advances have increased the amount of recorded temporal information. As a result, the sequence of networks describing changes occurring over time have been formalized as *temporal networks* (also known as time-varying or time-stamped) [15]. Examples of temporal networks include the functional brain networks [2, 5], social media interactions [36], financial markets [1] or politics [22].

One aspect of temporal network analysis is the discovery of community structures, which are groups of nodes that are more densely connected to each other than they are to the rest of the network [26]. Changes in the configuration of communities over time signals important turns in the evolution of the system. Real data networks are often observed to have communities with a hierarchical structure referred to as multi-scale communities [29]. Changes in the community structure over time might take place either at one scale or across all scales of the community structure. For this reason, there is interest in methods that are able to investigate communities at different “scales” over time [25, 26, 35].

Some recent approaches to community detection in temporal networks rely on a simple network aggregation procedure whereby all time networks are first collapsed into a single network. Afterward traditional algorithms for community detection can be used [34]. These methods, however, ignore valuable information about the evolution of the community structures over time. Other methods investigate each time network individually [1, 11, 17, 22], thus ignoring the dependence of community structures between neighbouring time points.

There exist methods that extend algorithms from one to multiple networks by using the multilayer formulation of a temporal network [18]. This special data structure allows inter-layer couplings between neighbouring time networks [18, 25]. One method, which is extended in this way, is the *modularity maximization* [26]. The modularity of a network is defined as the number of connections within a community compared to the expected number of such connections in an equivalent random network. The generalisation of the *modularity maximization* introduced in [25] overcomes the obstacles mentioned earlier by using the multilayer formulation. In this way it introduces a dependence between communities identified in one layer and connectivity patterns in other layers.

Modularity maximization is controlled by a resolution parameter γ , determining the size of the detected communities and supporting the detection of multi-scale communities. However, the range of parameter values must be manually selected. The importance of relevant scales is assessed using stability procedures, which compare the detected communities to those obtained from random networks [28, 30]. When dealing with real life data, communities at one or more scales can go undetected if appropriate parameter values are not selected. The modularity maximization has also been used to investigate the effect of constant inter-layer weights between consecutive time layers [3] on the behavior of community detection.

Other approaches to multi-scale community mining in monoplex networks have been proposed to address some of the issues experienced by the modularity maximization. The method in [35] relies on spectral graph wavelets [14] and

introduces a notion of scale in the network. These wavelets are thus used to obtain a local view of the network from each node. The clustering properties of the spectrum of the Laplacian in clustering [4, 12, 21] are used to construct a wavelet filter function which enables the spectral graph wavelets [14] to be sensitive to multi-scale communities. Contrary to the modularity maximization, this method is able to automatically select the range of scales to be investigated for existing communities. For a better understanding of the current paper we suggest the reader gets acquainted with articles [14, 35].

In this paper we extend spectral graph wavelets to temporal networks. For this extension we use the supra-Laplacian of the temporal network, which is defined as the Laplacian of the matrix representation of its multilayer formulation. A challenge we face here is the need to take into account the fundamental difference between within-layer and inter-layer edges when studying the spectral properties of the supra-Laplacian [9, 18, 33]. Although some studies explain the effect of different inter-layer weights over the eigenvalues of the supra-Laplacian [24, 31], there is no work related to the interpretation of the eigenvectors of the supra-Laplacian for clustering purposes.

Using Perturbation theory [6, 32], we argue that the eigenvectors corresponding to the smallest eigenvalues of the supra-Laplacian are a linear combination of the eigenvectors – corresponding to all zero eigenvalues – of the Laplacian matrices of the separate time layers. From spectral graph theory [7], it is known that an eigenvector corresponding to the zero eigenvalue of the Laplacian matrix is not informative of the community structure. For this reason, the eigenvectors of the supra-Laplacian matrix, which can be obtained as approximations to these eigenvectors, cannot be used to identify communities within the time layers, and larger eigenvalues should be sought. Using the above arguments as a stepping stone, we propose a novel Temporal Multi-Scale Community Detection (TMSCD) method, which extends the notion of spectral graph wavelets to temporal networks and automatically selects relevant scales at which multi-scale community partitions are obtained. The method uses the relevance of a newly identified eigenvalue of the supra-Laplacian, which captures the coarse description of communities prevalent over time.

In what follows, we first define the notation used throughout this paper in Sect. 2. Section 3 describes the method for multi-scale community detection in temporal networks which uses the spectral properties of the supra-Laplacian to identify relevant scale. In Sect. 4 we compare the performance of TMSCD to the modularity maximization [25]. Section 5 concludes this paper.

2 Notation

Let $G = (V, A)$ be an N -node network where V is the set of nodes and $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix with edge weights between pairs of nodes $\{A_{ij} | i, j \in \{1, 2, \dots, N\}\}$. We only consider undirected, adjacency matrices ($A_{ij} = A_{ji}$ for all i and j). The degree of a node i is $d_i = \sum_{j=1}^N A_{ij}$, and the

degree matrix D has d_1, d_2, \dots, d_N on its main diagonal. Network G is associated with the normalized Laplacian matrix $L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$.

The networks representing different time points in the *temporal network* are known as layers. We use the notation $G^t = (V, A^t)$ for layer t in the *temporal network* $\mathcal{T} = \{G^1, G^2, \dots, G^T\}$, which is the ordered sequence of networks for $t \in \{1, 2, \dots, T\}$ time points, and we denote node i in layer t by i_t . We work with *temporal network* in which each node is present in all layers. The multilayer framework of a *temporal network* considers a diagonal ordinal coupling of layers [2, 18, 25]. In essence, inter-layer weights exist only between corresponding nodes in neighboring time layers. We denote the inter-layer edge weight for node i between consecutive layers t and $t + 1$ by $\omega_i^{t,t+1} \in \mathbb{R}$. Else $\omega_i^{t,p} = 0$ for $p \neq t - 1, t + 1$.

This *temporal network* \mathcal{T} has an associated adjacency matrix \mathcal{A} of size $NT \times NT$ – the supra-Adjacency matrix. The time-dependent diagonal blocks of \mathcal{A} , $\mathcal{A}_{t,t}$, are the adjacency matrices A^t , and the off-diagonal blocks, $\mathcal{A}_{t,t+1}$, are the inter-layer weight matrices $W^{t,t+1} = \text{diag}(\omega_1^{t,t+1}, \omega_2^{t,t+1}, \dots, \omega_N^{t,t+1})$. Else $\mathcal{A}_{t,p}$ is a $N \times N$ zero matrix for $p \neq t - 1, t + 1$.

The within-layer degree of node i in layer G^t is $d_i^t := \sum_{j=1}^N A_{ij}^t$ while the multilayer node degree of node i in layer G^t is $\mathfrak{d}_i^t := d_i^t + \omega_i^{t,t-1} + \omega_i^{t,t+1}$. These define the degree matrix \mathcal{D} with diagonal entries $\mathcal{D} := \text{diag}(\mathfrak{d}_1^1, \mathfrak{d}_2^1, \dots, \mathfrak{d}_N^1, \mathfrak{d}_1^2, \dots, \mathfrak{d}_N^2, \dots, \mathfrak{d}_N^T)$. The *normalized supra-Laplacian* \mathcal{L} is computed as $\mathcal{L} = \mathcal{D}^{-\frac{1}{2}}(\mathcal{D} - \mathcal{A})\mathcal{D}^{-\frac{1}{2}}$.

3 Temporal Multi-scale Community Detection (TMSCD)

The proposed TMSCD method is a multilayer extension of the multi-scale community detection procedure via spectral graph wavelets developed in [35]. The advantage of this method is the automated selection of relevant scales at which community partitions are obtained. In Sect. 3.1 we define new inter-layer weights at each node adapted for community detection in temporal networks. In Sect. 3.2 we extend the definition of a wavelet at a node (in a monoplex network) to that for a wavelet at a node at a particular time layer. By construction, a wavelet associated to a node at a time layer is local in the whole *temporal network*. The wavelet is centred around this node and spreads on its neighbourhood, which consists of its neighbours in the current layer and the corresponding nodes in the neighbouring time layers. The larger the scale is, the larger the spanned neighbourhood is – more nodes in current layer and more nodes in neighbouring layers.

The most central part of our method is the construction of the wavelet filter function g . In Sect. 3.3 we use arguments from Perturbation theory to investigate the spectral properties of the supra-Laplacian matrix for community detection purposes, and propose a procedure for the selection of appropriate eigenvalues around which to center the wavelet filter function. In Sect. 3.4, we introduce the wavelet filter function based on a B -spline, we choose the parameters of this function, and define relevant scales for community investigation.

Finally, in Sect. 3.5 for any given scale, we use the wavelet of a node at a given time layer to cluster together nodes whose associated wavelets are correlated using an agglomerative connectivity-constrained clustering procedure which respects the time sequence of the temporal network.

3.1 Inter-layer Couplings $\omega_i^{t,t+1}$ for Community Detection

The choice of inter-layer weights $\omega_i^{t,t+1}$ is important - they control the ordinal coupling between time layers t and $t+1$ via the node i . We believe that inter-layer couplings should be strong enough to indicate similarity of a node's neighbourhood in two consecutive networks and indicate shared community structures over time. The main principle is, inter-layer weights should be strong enough to reflect on local topological similarity of nodes across layers, but they should not interfere with the within-layer structure.

Let the set of neighbours of node i in layer G^t be denoted by $\mathcal{N}_i^t := \{j_t : A_{ij}^t = 1\}$.

We introduce the inter-layer weight $\omega_i^{t,t+1}$ as follows:

$$\omega_i^{t,t+1} := \frac{|\mathcal{N}_i^t \cap \mathcal{N}_i^{t+1}|}{2}. \quad (1)$$

We refer to these inter-layer weights as LART-type since they were the basic ingredients of the LART algorithm [19]. The LART algorithm is a method for the detection of communities that are shared by either some or all the layers in a multilayer network. The algorithm is based on a random walk and the transition probabilities defining the random walk are allowed to depend on the local topological similarity between layers at any given node.

It can be derived that $\omega_i^{t,t+1} \leq \frac{\min(d_i^t, d_i^{t+1})}{2}$. From this follows that the multi-layer node degree of i_t is $d_i^t \leq 2d_i^t$. Thus at least half of the influence, which node i_t has over the properties of \mathcal{A} and therefore \mathcal{L} , comes from the connections of node i within layer t , rather than from the inter-layer weights $\omega_i^{t,t-1}$ and $\omega_i^{t,t+1}$.

3.2 Construction of Spectral Graph Wavelets for Temporal Networks

Upon obtaining matrices \mathcal{A} and \mathcal{L} , we construct the spectral graph wavelet transform and the corresponding wavelet basis using the spectral decomposition of \mathcal{L} as in [14, 35]. Let $\Lambda = \{\lambda_j\}_{j=1}^{NT}$ be the vector of eigenvalues of the supra-Laplacian \mathcal{L} satisfying $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{NT}$. Let $\chi = [\chi_1 | \chi_2 | \dots | \chi_{NT}]$ be the $NT \times NT$ matrix of column eigenvectors which correspond to those eigenvalues.

Denote by $\psi_{s,i}^t$ the wavelet at scale $s \in \mathbb{R}^+$ centred around node $i \in V$ at time layer t . The wavelets are generated by stretching a unique wavelet filter function $g(\cdot)$ by a scale parameters $s > 0$ in the network Fourier domain. The matrix representation of the stretched filter is

$$\mathcal{G}_s = \text{diag}(g(s\lambda_1), g(s\lambda_2), \dots, g(s\lambda_{NT}))$$

that is diagonal on the Fourier modes (the NT eigenvectors of \mathcal{L}). Hence the wavelet basis at scale s reads as

$$\Psi_s = (\psi_{s,1}^1 | \psi_{s,2}^1 | \dots | \psi_{s,N}^1 | \psi_{s,1}^2 | \dots | \psi_{s,N}^T) = \chi \mathcal{G}_s \chi^\top, \quad (2)$$

where $\psi_{s,i}^t$ is the wavelet at scale s centred around node i at the time point t . For a wavelet at scale s centered around node i at time point t , we have the relation $\psi_{s,i}^t = \chi \mathcal{G}_s \chi^\top \delta_{i,t}$, which is a column vector of size NT . Its value at each node j at time point p is given by $\psi_{s,i}^t(j, p)$.

3.3 Spectral Properties of the Supra-Laplacian Matrix for Community Detection Purposes

In our context, we interpret each G^t as disconnected components and the inter-layer weights as small perturbations. The resulting diagonal blocks of the supra-Laplacian, $\mathcal{L}_{t,t}$, are then perturbed versions of the corresponding Laplacian L^t . We use Davis-Kahan theorem from matrix Perturbation theory (p. 246 in [32] and p. 212 in [6]) discussed in [21] to justify the choice of an eigenvalue around which to center the wavelet filter function. According to the Davis-Kahan theorem, *some* of the first T perturbed eigenvectors of \mathcal{L} are very close to the linear space generated by the vectors $v_0^t \mathbb{1}_{G^t}$. Here v_0^t is the eigenvector corresponding to eigenvalue 0 of matrix L^t , while $\mathbb{1}_{G^t}$ is the NT zero-padded indicator vector, which has entries 1 at the node positions of layer G^t .

From spectral graph theory [7] it follows that the eigenvector v_0^t corresponding to the 0 eigenvalue of the normalized Laplacian matrix L^t (of the undirected connected network G^t) is not informative of the community structure, since it is equal to the squared node degrees, $D_t^{\frac{1}{2}}$. It follows that in the spectrum of the supra-Laplacian there exists a set of small eigenvalues λ , whose corresponding eigenvectors are *uninformative* for the community structure within the layers.

These eigenvalues and their corresponding eigenvectors can only be used to identify each time layer G^t as a separate layer. In fact, the smallest non-zero eigenvalue λ , whose eigenvector is *not spanned* by the set of eigenvectors v_0^t , is sensitive to within-layer connectivity patterns since it may appear as perturbation of the separate layers' Fiedler vectors used for clustering [7]. We center our wavelet filter function around this eigenvalue, denoted by λ^* , since λ^* is carrier of the coarse description of communities within time layers. We also use λ^* to automatically determine the range of scales s , for which relevant communities can be discovered.

Denote by $\bar{\Lambda}$ the set of smallest eigenvalues whose eigenvectors are well-approximated by the subspace of eigenvectors v_0^t . According to the Davis-Kahan theorem, the eigenvectors v corresponding to $\lambda \in \bar{\Lambda}$ satisfy

$$\min_{\{\alpha_t\}} \left\| v - \sum_{t=1}^T \alpha_t v_0^t \right\| \leq \varepsilon \quad (3)$$

for a small $\varepsilon > 0$. For the rest of the eigenvalues, the left hand side of this inequality is much larger than ε . Then eigenvalue λ^* is the first eigenvalue which

is a perturbation of the Fiedler vectors of the separate time layers, i.e. we have the equality

$$\lambda^* := \min \{ \lambda : \lambda \in \Lambda \setminus \bar{\Lambda} \}. \quad (4)$$

In practice, we discover the position of the eigenvalue λ^* by solving a series of regression problems: for each of the ordered eigenvectors of the supra-Laplacian $v_\tau = \chi_\tau$ ($\tau = 1, 2, \dots, NT$), we fit the multivariate regression $v_\tau = \sum_{t=1}^T \beta_t \alpha_t v_0^t + \varepsilon_\tau$. We select λ^* at the τ position for which $\|\varepsilon_\tau\| > 0.8$, where this bound was empirically selected. Since $\lambda^* \leq \lambda_{T+1}$, we have to solve at most $T+1$ regression problems in order to find the position of λ^* .

3.4 Graph Wavelet Filter g via B -Splines and Parameter Selection

We propose a new wavelet filter function g modeled as a cubic B -spline [8] with appropriately chosen knots. This function is not only smooth but also has a compact support. Namely, we put

$$g(y) := B_3(0, y_1, y_2, y_3, y_4; y) \quad (5)$$

where for the knots of the cubic B -spline B_3 we have

$$0 < y_1 < y_2 = y_3 < y_4 \quad (6)$$

and the function g is zero out of the interval $[0, y_4]$. By the properties of B -splines, $B_3 > 0$ for $y \in (0, y_4)$. As indicated, this spline function has a double knot at $y_2 = y_3$. Function g inherits the basic properties of B -splines [8], including good properties of the Fourier coefficients of g since $g(y)$ may be extended for $y < 0$ and $y > y_4$ periodically to belong to C^1 , which is important for the invertibility of the Fourier wavelet transforms. Other functions can further be pursued depending on the application at hand, and possible options have been reviewed in [20].

In the following we describe how to choose parameters y_1 , $y_2 = y_3$, and y_4 of the wavelet filter function g , and the range of scales s relevant for multi-scale communities within and across layers of the temporal network. Some of the arguments we make are the same as in [14, 35]. However, we adapt these to the nature of g and the aim of centering it around an appropriate eigenvalue.

First, the maximum scale s_{\max} is set so that the filter function $g(s_{\max}y)$ starts decaying as a power law only after $y = \lambda^*$, hence $\lambda^* s_{\max} = y_2 = y_3$. Second, we need to keep a part of the corresponding eigenvector χ_{λ^*} in the wavelets of every scale, so that all wavelets are sensitive to the large scale community structure within each time point. We propose as minimum scale s_{\min} the one for which $g(s_{\min}\lambda^*)$ becomes smaller than 1. Therefore, $s_{\min}\lambda^* = y_1$. We also impose that $g(s_{\min})$ spans at least the whole range of eigenvalues between 0 and 1 which implies $s_{\min} \times 1 = y_2$.

We require that the filter at the maximum scale be highly selective around λ^* . For this purpose all other eigenvalues and especially λ^{q+1} , where we have put $\lambda^* = \lambda^q$, have to be attenuated. Choosing an attenuation by a factor 10,

leads to $g(s_{\max} \lambda^q) = 10 (s_{\max} \lambda^{q+1})$. We thereby ensure that the filter at the maximum scale essentially keeps the information from χ_{λ^*} .

This argumentation gives us spectrum adapted equations for s_{\min} , s_{\max} :

$$s_{\min} = \frac{y_1}{\lambda^*}, \quad y_2 = y_3 = \frac{y_1}{\lambda^*}, \quad s_{\max} = \frac{y_1}{(\lambda^*)^2}, \quad (7)$$

where we see that y_1 has the unique effect of translating the scale boundaries s_{\min} and s_{\max} on the \mathbb{R}^+ axis. Therefore, y_1 can be safely fixed to 1, i.e. $y_1 = 1$. Finally, similar to the approach in [14,35] we choose a logarithmically spaced sampling of M scales between the scale boundaries s_{\min} and s_{\max} : $S = \{s_1 = s_{\min}, s_2, \dots, s_M = s_{\max}\}$.

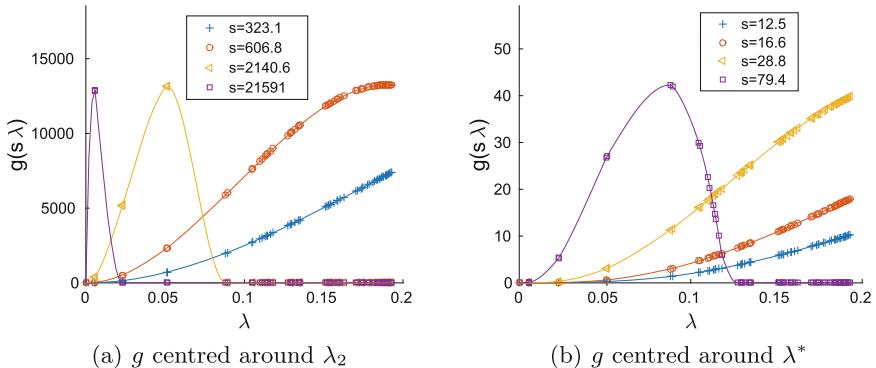


Fig. 1. *B*-spline based filter function g for four different scales s . The actual eigenvalues λ are obtained from a temporal multi-scale benchmark whose communities at large scales change over time (discussed in Sect. 4). The temporal network has $N = 640$ nodes for each of the $T = 33$ time layers with Sales-Pardo parameters $\rho = 1$ and $\bar{k} = 16$. In (a) filter g is centred around λ_2 as originally proposed in [35] for monoplex networks. In (b) the filter g is centred around λ^* , which is λ_{10} obtained as proposed in Sect. 3.3. In total 50 scales were obtained and the four visualized scales correspond to the 7th, 13th, 25th and 47th scale.

Rather than fixing the first parameter y_2 of filter g around the second eigenvalue λ_2 of the supra-Laplacian, we fix y_2 centred around λ^* . Thus we *attenuate* the role of the eigenvalues $\lambda_j \in \Lambda$ with $\lambda_j < \lambda^*$, since as we explained above the eigenvectors of these eigenvalues λ_j are not relevant for discovering community structures prevalent at each time point.

In Fig. 1 we visualize function g for a *temporal network* and compare the shape of g when it is centred around λ_2 and around λ^* – obtained as proposed. The eigenvalues were obtained from a multi-scale benchmark temporal network whose communities at large scales change over time.

3.5 Agglomerative Connectivity-Constrained Clustering and Detection of Stable Partitions

For small scales s , $\psi_{s,i}^t$ is localized around the direct neighbours of i in layer t and to few nodes in neighbouring time layers. With an increasing scale s , $\psi_{s,i}^t$ spreads to a larger neighbourhood which eventually becomes the whole multilayer network. Hence, we use $\psi_{s,i}^t$ as a feature vector for i_t at scale s .

Similar to [35], we determine the distance $D_s(i_t, j_p)$ between nodes i_t and j_p ($i, j = 1, 2, 3, \dots, N; t, p = 1, 2, 3, \dots, T$) at scale s using the correlation distance between wavelets $\psi_{s,i}^t$ and $\psi_{s,j}^p$. We speed up computations of the full spectrum of \mathcal{L} and all $D_s(i_t, j_p)$ using approximations proposed in [14, 35].

We cluster nodes into communities using distances $D_s(i_t, j_p)$ and an agglomerative connectivity-constrained clustering procedure [19, 27] with “average” linkage. In this way we respect the time-ordered structure of the temporal network since nodes in the same time layer or nodes across neighbouring time layers are considered first for merging. We obtain the partition at scale s , P_s , by cutting the resulting dendrogram at a height equal to the average of the maximal gaps of all the root-leaf paths of the dendrogram [35]. Repeating the above for all $s \in S$, we obtain the multi-scale set of partitions $\mathcal{P} = \{P_s\}_{s \in S}$. We calculate the stability $\gamma_a(s)$ of the partition at a given scale s using the approach outlined in [35].

4 Experimental Results

In this section we provide simulation experiments to measure the performance of the TMSCD method on two types of benchmarks for temporal networks in comparison to the performance of the modularity maximization (MM) method [25], for different resolution parameter values γ on the same set of benchmarks.

The first type of benchmark networks we use is a further contribution of the present work, since we identify three classes of temporal networks which may serve as *benchmarks* for multi-scale community detection on temporal networks. We construct these benchmarks as time-varying Sales-Pardo (SP) networks [29]. An SP has three scales of communities based on which the network is constructed using parameters ρ (quantifies how separated the three scales are) and \bar{k} (the average node degree that controls how dense the network is). For a given length of the temporal network T , we generate SP multi-scale community structures that merge and split over time. Based on these community structures, we simulate a time-ordered sequence of Sales-Pardo networks. The three classes are determined by the scale at which the change occurs: small scale (SSC), medium scale (MSC) or large scale (LSC) change over time. The second type of benchmarks, proposed in [13], have one “true” partition at each time point.

The performance of a given algorithm at a given scale (resolution) is measured as the maximum value of the adjusted rand index (ARI) [16] between the “true” partition of the benchmark and the partition P at this scale (resolution). Since the SP benchmarks have three true partitions corresponding to three different scales, we refer to the large (resp. medium, small) scale as LS (resp. MS, SS). We

also investigate the performance of TMSCD and MM on benchmarks produced using different values of ρ and \bar{k} .

For the TMSCD method, the instability $1 - \gamma_a$ at scale s is obtained as outlined in [35]. The smaller $1 - \gamma_a$, the more stable is the community partition for scale s . For the MM method, the instability for a resolution parameter γ is obtained as described in [10] and is measured by the normalized variation of information (VI) metric [23]. The smaller VI, the more stable is the community partition at resolution parameter γ .

4.1 Comparative Results on Temporal Benchmark Networks with Multi-scale Community Structure

Discussion on Effect of Inter-layer Weights on the Performance of TMSCD and MM. First, we illustrate the performance of the TMSCD and MM method on an SSC network with $T = 21$ time layers, where $\rho = 1$, $\bar{k} = 16$, and $N = 640$ at each time point. For both methods, we use different fixed inter-layer weights $\omega = 0.5, 1, 2, 5, 10$ and the LART-type inter-layer weight $\omega_i^{t,t+1}$ proposed in Sect. 3.1, which we refer to as $\omega = LART$. For TMSCD, we set $M = 50$ scales $s \in S$, and for MM we manually set 60 values of resolution parameters γ in the interval $[0.05, 40]$ such that there are more values in the interval $[0.05, 1]$. For both methods we use 20 repetitions to obtain instability $\gamma_a(s)$ at scale s and $VI(\gamma)$ at each resolution γ .

The results of TMSCD and MM on a realization of the SSC and the instabilities $1 - \gamma_a$ versus scale s (VI versus parameter γ) for different weights ω are presented in Fig. 2. Results for realization of the MSC and LSC are omitted in this paper but the plots are similar and conclusions are identical.

Overall, the TMSCD recovers perfectly communities at all three scales and inter-layer weights ω have almost no effect over the results. For small ω ($\omega = 0.5, 1, 2$) instability is high, but for $\omega = LART$ and $\omega = 5, 10$ the associated partitions of scales with low instability correspond to the true partitions. For large ω ($\omega = 5, 10$ and $\omega = LART$) a fourth stable scale appears at the smallest s . This is stable for $\omega = 5, 10$ but unstable for $\omega = LART$, which signifies the importance of carefully selected weights. MM recovers perfectly communities at LS and MS, but there is increased variability at recovering communities at SS for an increasing inter-layer weight ω . The instability of MM is not as sensitive as the one that is used for TMSCD.

To conclude, using $\omega = LART$ inter-layer weights appears to provide us with low instability only at the true partitions. This includes a higher instability at the fourth scale which appears for larger ω ($\omega = 10$) - the partition at this scale is formed for N communities, and each community is formed by the set of nodes $\{i_t : t = 1, 2, \dots, T\}$. As discussed in Sect. 3.1, this phenomenon is a result of larger inter-layer weights, which affect more the properties of a node i at layer t than its within-layer connections which have an average node degree $\bar{k} = 16$. In contrast, the LART weights, in the range $[0, 8]$, follow a bell-shaped distribution centred around 4. Thus they do not interfere with the within-layer connections and support the community detection process over time.

On the other hand, the instability procedure for MM is not as sensitive. In the case of real data it would be challenging to select parameters γ for which to investigate community partitions.

Discussion on Overall Performance of TMSCD and MM. We compare TMSCD and MM for different sets of parameters $\rho = 1, 2$ and $\bar{k} = 11, 13, 15, 17, 19, 21$, where we set LART-type inter-layer weights. We compare the obtained communities to the ground truth for 50 realizations of the SSC (Fig. 3(a)), MSC (Fig. 3(b)) and LSC (Fig. 3(c)). For each combination (ρ, \bar{k}) , the large scale rate (LSR) (resp. medium (MSR) and small (SSR) scale rates) indicates the success rate of the communities found by TMSCD and MM being compared to the large (resp. medium, small) scale ground truth community structure. The success rate is the average over the top five adjusted rand index (ARI) values over all scales s or parameter values γ .

For $\rho = 1$, both methods perform equally well in all three cases with almost full recovery of communities at all scales. In some cases, TMSCD has slight advantage of recovering MS and LS communities. For $\rho = 2$, the performance of both methods decreases for small \bar{k} . Both methods perform equally well at recovering MS communities, but we can note that TMSCD performs better at recovering LS communities for larger \bar{k} .

Overall TMSCD performs slightly better than MM and has much smaller deviation in the final results. In general, uncovering communities when $\rho = 2$ is harder since ρ controls how separated are the communities at the three scales. When ρ is larger, the separation of the communities is not as clear, so SS communities cannot be distinguished easily. Furthermore, we note that when \bar{k} is small nodes have fewer edges and it is difficult for both methods to uncover SS communities since they fade in the MS communities.

4.2 Comparative Results on Benchmarks with One True Partition

We compare the performance of TMSCD and MM on the *Grow*, *Merge* and *Mixed* benchmark networks proposed in [13], with default model parameters, and we set $T = 100$ and $N = 128$. We produce 100 realizations of each benchmark. Parameters of TMSCD and MM are set as in Sect. 4.1, where we use LART-type inter-layer weights. The success rate of each realization of a benchmark is the average over the top five adjusted rand index (ARI) values over all scales s or parameter values γ . The results are summarized in Table 1.

Both methods perform equally well for the Grow model. In the Mixed model case, TMSCD has better performance with lower variability. In the Merge model case, TMSCD performs much better than MM but with larger variability in the results. The Merge model is challenging for both MM and TMSCD. This is caused by the nature of the communities: when two communities are separate they exist at a smaller scale, but when they merge they exist at a larger scale.

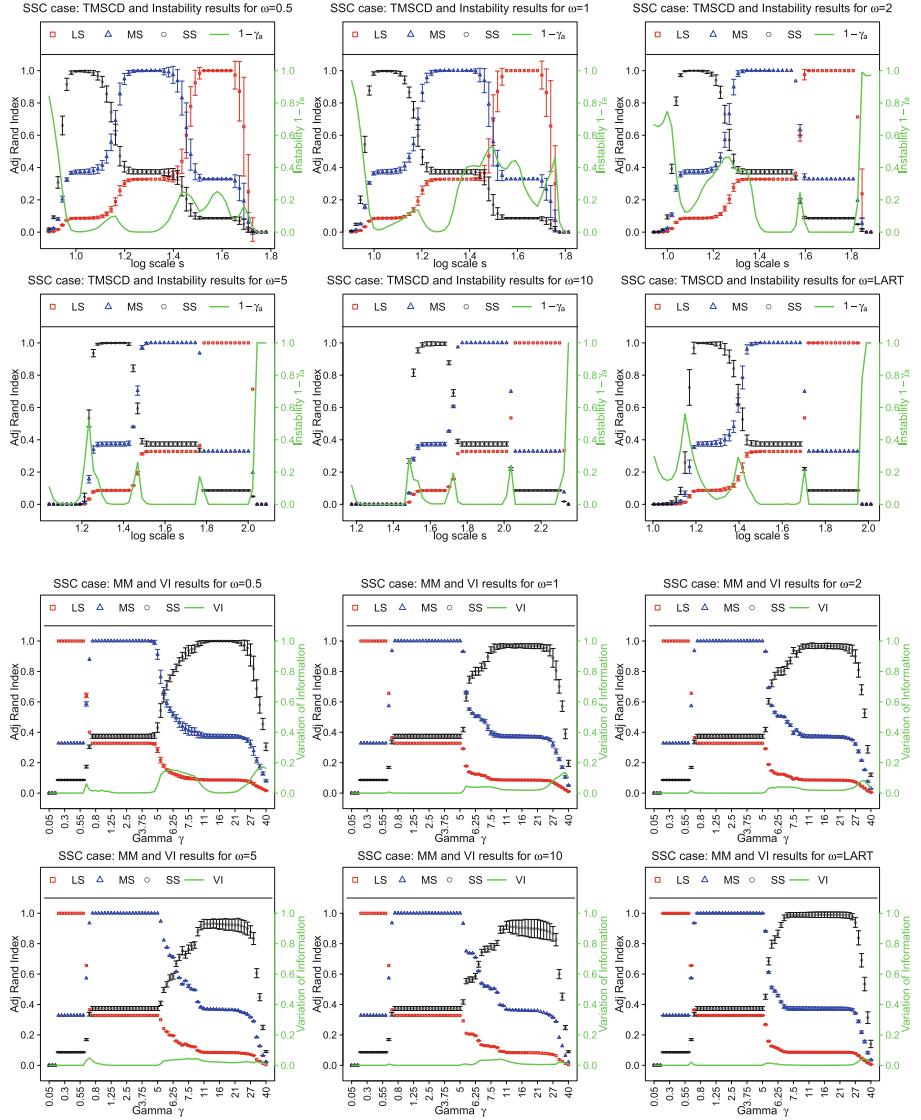
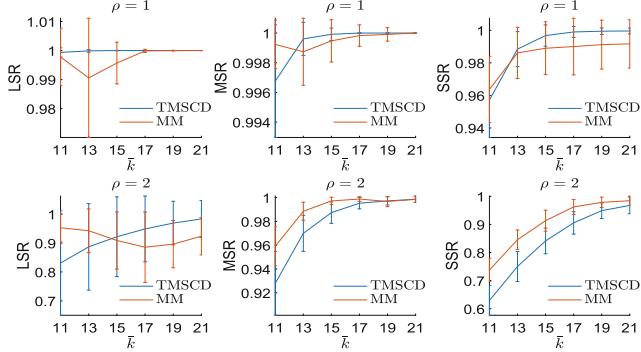
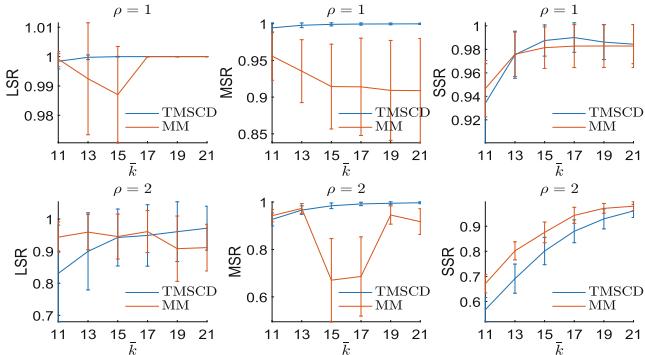


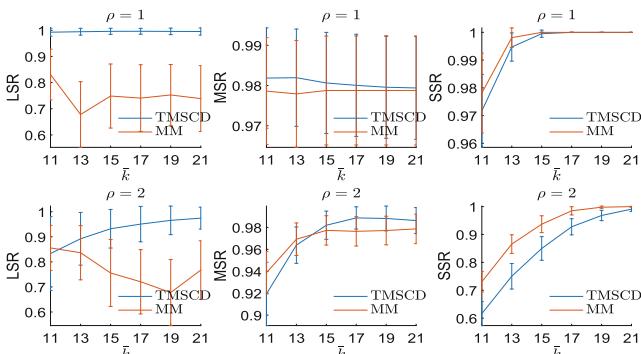
Fig. 2. TMSCD (top) and MM (bottom) results for SSC multi-scale benchmark network. Each pair of plots corresponds to different inter-layer weights ω . First, we plot the results of TMSCD and MM on a realization of SSC. Each scale outputs a partition for all nodes across all time points. For each scale s , we plot the similarity with the small (SS) (medium (MS), large (LS)) theoretical scale, computed as the average over all time points including std.dev. error bars. We observe scales where the exact small (resp. medium, large) scale theoretical partition is uncovered. Second, for TMSCD we plot instability $1 - \gamma_a$ versus scale s ; for MM we plot variation of information (VI) versus parameter γ . The associated partitions of scales with low instability corresponding to the theoretical partitions.



(a) Results for SSC temporal benchmark



(b) Results for MSC temporal benchmark



(c) Results for LSC temporal benchmark

Fig. 3. Comparison between the LSR, MSR and SSR values obtained for the TMSCD and MM multi-scale community mining methods on (a) SSC, (b) MSC and (c) LSC temporal benchmark network for different parameters: left (resp. right) column for $\rho = 1$ (resp. $\rho = 2$) and different values of \bar{k} . We plot the average and the \pm one standard deviation over the five best results for each of the 50 realizations of each type of network for each set of parameters.

Table 1. TMSCD and MM results for 50 realizations of the Grow, Merge and Mixed model. Each entry is the mean over all realizations \pm one standard deviation.

	Grow	Merge	Mixed
TMSCD	1.0000 ± 0.0000	0.8700 ± 0.1981	0.9467 ± 0.1038
MM	0.9975 ± 0.0088	0.6887 ± 0.1302	0.8443 ± 0.1412

5 Discussion and Conclusion

The work in this paper is motivated by the need to develop new methods for multi-scale community detection in temporal networks with automatic selection of relevant scales. The modularity maximization [25] achieves excellent results at detecting such communities but it lacks the flexibility of automatically selecting resolution parameter ranges relevant to the prevalent community structures over time. We have used results from Perturbation theory to interpret inter-layer weights as perturbations between time layers, and thus we identify the set of eigenvectors of the supra-Laplacian that are perturbations of the separate layers' Fiedler vectors. These can be used for detecting communities prevalent over time.

This result gives a completely new point of view on *temporal networks*. To design the TMSCD method, we reconsidered the role of the Fiedler vector in community detection for *temporal networks*. Indeed, the eigenvectors of \mathcal{L} (corresponding to the smallest eigenvalues) represent all time-layers as separate communities. Hence, we cannot use them for the detection of communities prevalent over time. We successfully attenuated the influence of these small eigenvalues in the process of community detection, by properly constructing the wavelet filter function g of the spectral graph wavelets. An important step in our algorithm was the identification of the uninformative small eigenvalues.

Using simulated data, we have demonstrated that TMSCD method performs equally well compared to the modularity maximization method [25]. There are two main advantages to using TMSCD. First, of utmost importance is the automatic selection of scales at which wavelets should be obtained and which encompass all relevant within-layer and inter-layer communities. Second, the proposed LART-type inter-layer weights $\omega_i^{t,t+1}$ lead to the best results in terms of balance between uncovering multi-scale communities and the stability of those communities at the relevant scales. The stability procedure used by TMSCD is more sensitive than the modularity maximization one. This is an advantage when handling real life data sets where the true scales are not known and a reliable indicator for stable partitions is required. The supremacy of the LART-type inter-layer weights [19] over using fixed ones indicates the advantage of using adaptable inter-layer weights that reflect the similarity of nodes across layers.

Given the above results, TMSCD would be an ideal tool for applications to neuroscience and social network analysis.

Acknowledgments. ZK acknowledges partial support by the Engineering and Physical Sciences Research Council (EPSRC) and by grant no. I02/19 of Bulgarian NSF.

References

1. Aynaud, T., Guillaume, J.L.: Static community detection algorithms for evolving networks. In: 2010 Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), pp. 513–519. IEEE (2010)
2. Bassett, D.S., Wymbs, N.F., Porter, M.A., Mucha, P.J., Carlson, J.M., Grafton, S.T.: Dynamic reconfiguration of human brain networks during learning. *Proc. Natl. Acad. Sci. U. S. A.* **108**(18), 7641–7646 (2011)
3. Bazzi, M., Porter, M.A., Williams, S., McDonald, M., Fenn, D.J., Howison, S.D.: Community detection in temporal multilayer networks, and its application to correlation networks. *Multiscale Model. Simul.* **14**(1), 1–41 (2014)
4. Bertrand, A., Moonen, M.: Seeing the bigger picture: how nodes can learn their place within a complex ad hoc network topology. *IEEE Signal Process. Mag.* **30**(3), 71–82 (2013)
5. Betzel, R.F., Griffa, A., Avena-Koenigsberger, A., Goni, J., Thiran, J.P., Hagmann, P., Sporns, O.: Multi-scale community organization of the human structural connectome and its relationship with resting-state functional connectivity. *Netw. Sci.* **1**(03), 353–373 (2014)
6. Bhatia, R.: Matrix Analysis. Graduate Texts in Mathematics, vol. 169. Springer, Heidelberg (1997). <https://doi.org/10.1007/978-1-4612-0653-8>
7. Chung, F.: Spectral Graph Theory. CBMS (1996)
8. De Boor, C.: A Practical Guide to Splines. Springer, Heidelberg (2001)
9. De Domenico, M., Solé-Ribalta, A., Cozzo, E., Kivelä, M., Moreno, Y., Porter, M.A., Gómez, S., Arenas, A.: Mathematical formulation of multilayer networks. *Phys. Rev. X* **3**(4), 041022 (2013)
10. Delvenne, J.C., Yaliraki, S.N., Barahona, M.: Stability of graph communities across time scales. *Proc. Natl. Acad. Sci. U. S. A.* **107**(29), 12755–12760 (2010)
11. Fenn, D.J., Porter, M.A., McDonald, M., Williams, S., Johnson, N.F., Jones, N.S.: Dynamic communities in multichannel data: an application to the foreign exchange market during the 2007–2008 credit crisis. *Chaos* **19**, 033119 (2009)
12. Fortunato, S.: Community detection in graphs. *Phys. Rep.* **486**(3–5), 75–174 (2010)
13. Granell, C., Darst, R.K., Arenas, A., Fortunato, S., Gómez, S.: Benchmark model to assess community structure in evolving networks. *Phys. Rev. E. Stat. Nonlin. Soft Matter Phys.* **92**(1), 012805 (2015)
14. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.* **30**(2), 129–150 (2011)
15. Holme, P., Saramäki, J.: Temporal networks. *Phys. Rep.* **519**(3), 97–125 (2012)
16. Hubert, L., Arabie, P.: Comparing partitions. *J. Classif.* **2**(1), 193–218 (1985)
17. Lee, J.D., Maggioni, M.: Multiscale analysis of time series of graphs. In: International Conference on Sampling Theory and Applications (2011)
18. Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A.: Multilayer networks. *J. Complex Netw.* **2**(3), 203–271 (2014)
19. Kuncheva, Z., Montana, G.: Community detection in multiplex networks using locally adaptive random walks. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015, pp. 1308–1315. ACM Press, New York, August 2015
20. Leonardi, N., Van De Ville, D.: Tight wavelet frames on multislice graphs. *IEEE Trans. Signal Process.* **61**(13), 3357–3367 (2013)

21. Luxburg, U.V.: A tutorial on spectral clustering. Technical report, Max Planck Institute (2007)
22. Macon, K.T., Mucha, P.J., Porter, M.A.: Community structure in the united nations general assembly. *Phys. A Stat. Mech. Appl.* **391**(1), 343–361 (2012)
23. Meila, M.: Comparing clusterings - an information based distance. *J. Multivar. Anal.* **98**(5), 873–895 (2007)
24. Moreno, Y., Arenas, A.: Diffusion dynamics on multiplex networks. *Phys. Rev. Lett.* 1–6 (2013)
25. Mucha, P.J., Richardson, T., Macon, K., Porter, M.A., Onnela, J.P.: Community structure in time-dependent, multiscale, and multiplex networks. *Science* **328**, 876–878 (2010)
26. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.* **103**(23), 8577–8582 (2006)
27. Pons, P., Latapy, M.: Computing communities in large networks using random walks. *J. Graph Algorithms Appl.* **10**(2), 1910218 (2006)
28. Lambiotte, R.: Multi-scale modularity in complex networks. In: 2010 Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), pp. 546–553 (2010)
29. Sales-Pardo, M., Guimerà, R., Moreira, A.A., Amaral, L.A.N.: Extracting the hierarchical organization of complex systems. *Proc. Natl. Acad. Sci. U.S.A.* **104**(39), 15224–15229 (2007)
30. Schaub, M.T., Delvenne, J.C., Yaliraki, S.N., Barahona, M.: Markov dynamics as a zooming lens for multiscale community detection: non clique-like communities and the field-of-view limit. *PLoS ONE* **7**(2), e32210 (2012)
31. Sol, A., Domenico, M.D., Kouvaris, N.E.: Spectral properties of the Laplacian of multiplex networks. *Phys. Rev. E* **88**(3), 032807 (2013)
32. Stewart, G.W., Sun, J.G.: *Matrix Perturbation Theory*. Academic Press, New York (1990)
33. Taylor, D., Myers, S.A., Clauset, A., Porter, M.A., Mucha, P.J.: Eigenvector-based centrality measures for temporal networks. arxiv Prepr., p. 34, July 2015
34. Traud, A.L., Mucha, P.J., Porter, M.A.: Social structure of Facebook networks. *Phys. A Stat. Mech. Appl.* **391**(16), 4165–4180 (2011)
35. Tremblay, N., Borgnat, P.: Graph wavelets for multiscale community mining. *IEEE Trans. Signal Process.* **62**(20), 5227–5239 (2014)
36. Zhao, X., Sala, A., Wilson, C., Wang, X., Gaito, S., Zheng, H., Zhao, B.Y.: Multi-scale dynamics in a massive online social network. In: Proceedings of the 12th ACM SIGKDD Internet Measurement Conference, pp. 171–184, May 2012

Influence Maximization-Based Event Organization on Social Networks

Cheng-Te Li^(✉)

National Cheng Kung University, Tainan, Taiwan
chengte@mail.ncku.edu.tw

Abstract. This paper is an ongoing work, and was presented as “Lightening Talk” in the DyNo workshop held at ECML/PKDD 2017. Online event-based social services allow users to organize social events by specifying the themes, and invite friends to participate social events. While the event information can be spread over the social network, it is expected that by certain communication between event hosts, users interested in the event themes can be as many as possible. In this work, by combining the ideas of team formation and influence maximization, we formulate a novel research problem, *Influential Team Formation* (ITF), to facilitate the organization of social events. Given a set L of required labels to describe the event topics, a social network, and the size k of the host team, ITF is to find a k -node set S that satisfying L and maximizing the *Influence-Cost Ratio* (i.e., the influence spread per communication cost between team members). Since ITF is proved to be NP-hard, we develop two greedy algorithms and one heuristic method to solve it. Extensive experiments conducted on Facebook and Google+ datasets exhibit the effectiveness and efficiency of the proposed methods. In addition, by employing the real event participation data in Meetup, we show that ITF with the proposed solutions is able to predict organizers of influential events.

Keywords: Event organization · Team formation
Influence maximization · Influential Team Formation
Social networks

1 Introduction

Team formation [6] in a social network is to find a set of experts such that not only a set of required labels is covered but also team members have lower communication cost with one another (i.e., well-connected in the underlying network). It is apparent that team formation can be applied to many real applications, such as searching for a group of employers to execute a project in a company, and composing an activity group for a cocktail party with particular themes. However, team formation techniques [1, 4, 7] are not applicable for organizing *Influential Events* in event-based social services (e.g. Meetup¹, Plancast², and

¹ www.meetup.com.

² www.plancast.com.

Facebook Events). Here we consider influential event organization is to find a set of persons who are interested in the themes of an event, have better social interactions (i.e., lower communication cost), and can attract a large number of people to participate in the event. It is common and realistic to organize influential events. The real-world scenarios on the demand of influential events may include organizing technical conferences, fund raising for earthquake victims, and initiating anti-nuclear campaign. In such scenarios, people attempt to maximize the number of participants since more participants mean a success of the events. One may think *Social Influence Maximization* [5], which aims at finding a set of seeding users such that the number of influenced users can be maximized, seems to be a solution. However, influence maximization techniques [2, 3, 9] are not applicable for influential event organization because they consider neither the set of required labels, nor the communication between the selected seed nodes.

This work proposes a novel problem, *Influential Team Formation* (ITF), in a social network. Given a set of required labels and the size k of the team, the ITF problem is to find a set S of nodes such that (a) the query label set is covered by the discovered k -node set S , and (b) the *influence-cost ratio* of nodes in S is maximized. We propose the Influence-Cost Ratio (ICR) to quantify the influence spread of the selected k nodes per communication cost. ICR of a node set S is defined as $ICR(S) = \frac{\sigma(S)}{c(S)}$, where influence spread $\sigma(S)$ is the expected number of nodes activated by S while the communication cost $c(S)$ is the sum of all-pair shortest path lengths between nodes in S . A team can derive a higher ICR if the team members can lead to higher influence spread and are well-connected. The ITF problem is challenging since maximizing influence spread contradicts minimizing communication cost. Influence maximization tends to select well-separated nodes because their activated nodes can have less overlapping. But team formation prefers well-connected nodes since they can produce lower communication cost.

It is worthwhile to note that a team is a task-oriented group whose team members not only possess some skill labels to deal with the task, but also well collaborate with each other. Therefore, the team formation problem asks for a set of required skill labels as the input, and expects that the discovered team members are equipped with some of the required skill labels and have good communication among them. Since we aim at forming influential “teams”, the selected team members (i.e., seeds) need to rely on a required set of skill labels and be well-connected to have good communication. In addition, “influential” teams also require the team members to be influential, i.e., team members should lead to higher influence spread in the social network. Consequently, the proposed ITF problem is a combination of team formation and influence maximization.

We create an example, as shown in Fig. 1, to exhibit the differences among team formation (TF), influence maximization (IM), and the proposed ITF. This example assumes the set of required labels is $\{a, b, c, d, e\}$ and $k = 3$. TF may select the set $S_{TF} = \{v_1, v_2, v_3\}$ since they cover more required labels and are well-connected. $ICR_{TF} = \frac{7}{3}$. IM will select the set $S_{IM} = \{v_1, v_4, v_6\}$ because

they can lead to highest influence spread. $ICR_{IM} = \frac{11}{5}$. ITF will find the set $S_{ITF} = \{v_1, v_5, v_6\}$ that leads to the highest $ICR_{ITF} = \frac{10}{3}$. It is because not only the union of the activated sets of v_1 , v_5 , and v_6 leads to the largest activated set (i.e., $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9, v_{14}, v_{15}\}$), but also v_1 , v_5 , and v_6 are interconnected with a triangle structure in the network.

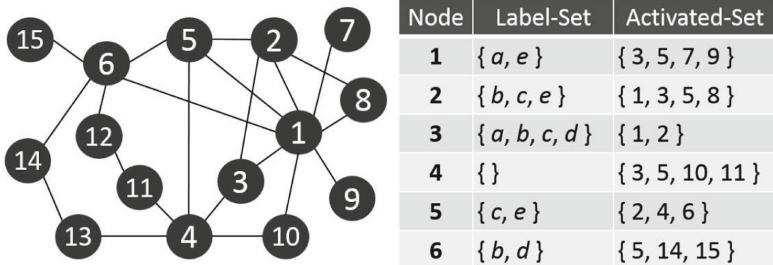


Fig. 1. A toy example of a social network (left), and a table (right) that describes the set of required labels possessed by each node and the set of nodes activated by each node. Note that a subset of nodes is shown in the table. Nodes except for v_1 to v_6 do not contain any required label.

In this talk, we will present the ITF problem under the *Independent Cascade* (IC) model. In order to solve the ITF problem. We propose a greedy algorithm with quality guarantee. While the greedy solution is effective but very inefficient, we further develop two greedy methods: *ICR Greedy* (ICR-Greedy) and *Mixed Influence-Cost Greedy* (M-Greedy), and one heuristic method: *Similar Influence Search* (SimIS). ICR-Greedy iteratively selects nodes with highest marginal gain of ICR scores. M-Greedy combines the NewGreedy IM method [3] with the original TF algorithm [6] in an interweaving manner. SimIS integrates Group-PageRank [8] with a best-first search in the social network. To validate the proposed methods, we have simulation-based and prediction-based experiments. The simulation-based experiments conducted on two real social network datasets, Facebook and Google+, and the results show both M-Greedy and SimIS can generate the highest ICR scores with satisfying time efficiency. The prediction-based experiments are conducted using the real event participation data of the event-based social service Meetup. The goal is to validate whether ITF with the proposed solution can truly identify the organizers of influential events based on the required labels of the given event and the social network. The results exhibit satisfying accuracy.

References

1. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: Proceedings of ACM International Conference on World Wide Web, WWW 2012, pp. 839–848 (2012)

2. Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2010, pp. 1029–1038 (2010)
3. Chen, W., Wang, Y., Yang, S.: Efficient influence maximization in social networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 199–208 (2009)
4. Kargar, M., An, A.: Discovering top- k teams of experts with/without a leader in social networks. In: Proceedings of ACM International Conference on Information and Knowledge Management, CIKM 2011, pp. 985–994 (2011)
5. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 137–146 (2003)
6. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 467–476 (2009)
7. Li, C.-T., Shan, M.-K., Lin, S.-D.: On team formation with expertise query in collaborative social networks. *Knowl. Inf. Syst.* **42**(2), 441–463 (2015)
8. Liu, Q., Xiang, B., Chen, E., Xiong, H., Tang, F., Yu, J.X.: Influence maximization over large-scale social networks: a bounded linear approach. In: Proceedings of ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, pp. 171–180 (2014)
9. Ohsaka, N., Akiba, T., Yoshida, Y., Kawarabayashi, K.-I.: Fast and accurate influence maximization on large networks with pruned Monte-Carlo simulations. In: Proceedings of AAAI Conference on Artificial Intelligence, AAAI 2014, pp. 138–144 (2014)

Author Index

- Allard, Tristan 10
Baesens, Bart 122
Bagheri, Mehrdad 76
Benkhelif, Tarek 36
Bioglio, Livio 3
Blanke, Tobias 110
Bouadi, Tassadit 10
Clérot, Fabrice 36
Cote, Mark 110
De Salve, Andrea 51
De Weerdt, Jochen 122
Duguépéroux, Joris 10
Fessant, Françoise 36
Fornacciari, Paolo 98
Giannotti, Fosca 64
Greenway, Giles 110
Guidi, Barbara 98
Hollmén, Jaakko 76
Kuncheva, Zhana 139
Lemahieu, Wilfried 122
Li, Cheng-Te 155
Manco, Giuseppe 23
Mitrović, Sandra 122
Montana, Giovanni 139
Mordonini, Monica 98
Mori, Paolo 51
Nanni, Mirco 64
Orlandini, Jacopo 98
Pappalardo, Luca 17
Pellungrini, Roberto 17
Pensa, Ruggero Gaetano 3
Pirrò, Giuseppe 23
Pratesi, Francesca 17
Pybus, Jennifer 110
Raschia, Guillaume 36
Ricci, Laura 51
Rinne, Mikko 76
Sani, Laura 98
Sans, Virginie 10
Tolvanen, Tuukka 76
Tomaiuolo, Michele 98
Trasarti, Roberto 64