PaperFree检测报告简明打印版

相似度：3.11%
编　号：DFVABHWCCGXZWLDJ
标　题：基于知识库的海量异构数据集成系统的设计与实现
作　者：-
长　度：23327字符
时　间：2017-06-20 14:21:16
比对库：中国学位论文全文数据库；中国学术期刊数据库；中国重要会议论文全文数据库；英文论文全文数据库；互联网资源；自建比对库

相似资源列表(学术期刊，学位论文，会议论文，英文论文等本地数据库资源)
1. 相似度：0.16%　篇名：《基于降维的输人情景化多维信息推荐研究》
来源：《情报理论与实践》　年份：2013　作者：杨君
2. 相似度：0.14%　篇名：《内存计算和列存储在商务智能中的应用分析》
来源：《中小企业管理与科技》　年份：2013　作者：周春娜
3. 相似度：0.09%　篇名：《常用算法时间复杂度的计算方法》
来源：《科技信息》　年份：2011　作者：殷超
4. 相似度：0.07%　篇名：《基于直线拟合算法的PET瓶瓶盖检测方法》
来源：《计算机应用研究》　年份：2011　作者：郑云峰
5. 相似度：0.07%　篇名：《基于q-gram的字符串相似性查询研究》
来源：《现代计算机：上下旬》　年份：2014　作者：米琳
6. 相似度：0.07%　篇名：《激励、整合与中介：知识库推广的可持续实践》
来源：《图书情报工作动态》　年份：2013　作者：陆彩女
7. 相似度：0.06%　篇名：《基于运动相似性的帧率提升算法》
来源：《电视技术》　年份：2012　作者：毛韧
8. 相似度：0.06%　篇名：《基于《中图法》的语义本体相似度技术研究》
来源：《情报科学》　年份：2013　作者：张瑾
9. 相似度：0.06%　篇名：《WordNet中基于IC的概念语义相似度算法》
来源：《计算机工程》　年份：2011　作者：王艳娜
10. 相似度：0.05%　篇名：《语义查询扩展在虚拟参考咨询知识库中的应用——以上海图书馆为例》
来源：《图书馆杂志》　年份：2013　作者：郭金龙
11. 相似度：0.05%　篇名：《解读大数据时代下档案管理的价值提升》
来源：《理论观察》　年份：2014　作者：张欣
12. 相似度：0.05%　篇名：《基于大数据的CDRS知识整合研究框架构建》
来源：《图书馆学研究》　年份：2013　作者：崔慧红
13. 相似度：0.05%　篇名：《高校信息系统集成的设计和实现》
来源：《辽宁中医药大学学报》　年份：2013　作者：潘旭
14. 相似度：0.05%　篇名：《提高审计数据分析的质量的对策》
来源：《管理工程师》　年份：2013　作者：李巍
15. 相似度：0.04%　篇名：《基于深度图像信息的视觉目标自遮挡检测方法研究》
来源：《燕山大学硕士论文》　年份：2010　作者：张煜婕
16. 相似度：0.04%　篇名：《基于多主体建模的学术社区知识共享行为仿真分析》
来源：《情报杂志》　年份：2013　作者：徐美凤
17. 相似度：0.04%　篇名：《基于领域本体的专题库构建——以中华烹饪文化知识库为例》
来源：《现代图书情报技术》　年份：2013　作者：许鑫
18. 相似度：0.04%　篇名：《供应商库存管理在半导体行业的实施》
来源：《物流工程与管理》　年份：2014　作者：夏荣忠

相似资源列表(百度文库，豆丁文库，博客，新闻网站等互联网资源)
1. 相似度：0.52%　标题：《How to Track International Flights in the Air | eHow》

来源：http://www.ehow.com/how_7363428_track-international-flights-air.html
2. 相似度：0.22% 标题：《【刘知远】知识图谱——机器大脑中的知识库》
来源：http://www.360doc.com/content/15/0801/10/22146031_488701821.shtml
3. 相似度：0.18% 标题：《Spring integration学习总结 - takeiteasy2009的专栏 - 博客频道 ...》
来源：http://blog.csdn.net/takeiteasy2009/article/details/8556394
4. 相似度：0.09% 标题：《Spring Integration集成框架之Message-Channel - 震秦的个人空间》
来源：https://my.oschina.net/zhzhenqin/blog/86586
5. 相似度：0.07% 标题：《网络维护人员需求报告》
来源：http://www.xielw.cn/2016/gongzuobaogao_1115/189351.html
6. 相似度：0.06% 标题：《基于KNN多要素中文文本分类的研究.pdf》
来源：http://max.book118.com/html/2016/0703/47199327.shtm
7. 相似度：0.05% 标题：《因子分析与其他分析的概念与区别 – MATLAB中文论坛》
来源：http://www.ilovematlab.cn/thread-16092-1-1.html
8. 相似度：0.05% 标题：《数据挖掘技术(四)——聚类_人生的悲哀_新浪博客》
来源：http://blog.sina.com.cn/s/blog_6002b97001014nja.html
9. 相似度：0.05% 标题：《“南海一号” 试捞20年沉浮录》
来源：http://art.people.com.cn/n/2013/1205/c206244-23750082.html
10. 相似度：0.04% 标题：《对三种典型分布式任务分配算法的分析_百度文库》
来源：
http://wenku.baidu.com/link?url=Qcu5Zo7dUPz_50CuGz4DPjor4jVP2MWwQ80fyQz6GkpnjbXVyqfBDx
11. 相似度：0.04% 标题：《管理信息系统习题答案(1--6)_百度文库》
来源：http://wenku.baidu.com/view/07e1e6f8aef8941ea76e05ed.html
12. 相似度：0.04% 标题：《基于语义连贯性实现主题挖掘和分类- 千与的专栏- 博客频道- CSDN....》
来源：http://blog.csdn.net/shirdrn/article/details/7076505

全文简明报告

摘 要

作为数据采集和分析的一个关键步骤,数据集成被应用于多个方面,诸如数据清洗、生物信息分析、模式识别等等。在当下的大数据时代,对于多数据源的集成往往很难的直接获得描述所有数据的全局模式,因此数据集成领域一个较为重要的课题就是全局模式的产生,而这一问题直接影响着数据集成后续步骤的效果。通过研究,我们分析了广泛数据源中属性名的表达形式以及彼此之间的关联和差异,提出了ED Join和Semantic Join两种用来集成相同属性的不同表达形式。经过实验,以上算法能够较高效、准确的进行模式集成,并依据此核心算法开发了基于知识库的海量异构数据集成系统。

关键词:信息集成;数据集成;模式匹配;知识库

Abstract

As the fundamental phrase of collecting and analyzing data, data integration is used in many applications, such as data cleaning, bioinformatics and pattern recognition. In big data era, one of the major problems of data integration is to obtain the global schema of data sources since the global schema could be hardly derived from massive data sources directly. In this paper, we attempt to solve such schema integration problem. We consider the representation difference of attribute names in various data sources and propose ED Join and Semantic Join algorithms to integrate attributes with different representations. Extensive experimental results demonstrate that the proposed algorithms could integrate schemas efficiently and effectively. Based on this, we develop knowledge-based massive data integration system.

Keywords: Information integration, Data integration,

Schema mapping, Knowledge base

第1章 绪论

本章主要介绍项目的来源、背景以及开发意义。由于此项目以创新性科研内容为主,本章还会简要说明现阶段国内外相关领域的最近研究进展。

1.1 课题背景

本项目来源于上海骇咕赛信息科技有限公司大数据平台数据集成子系统的最新需求。公司长期专注于企业级高并发高可用性技术解决方案和数据分析、数据挖掘领域,主要产品有高并发高可用性分布式缓存集群系统,该项目设计为大型计算集群、为分布式计算服务提供热缓存数据交换。在此系统之上,针对于海量数据整合、处理、分析、学习的需求,公司力主架构于该分布式系统,实现简便、高效的海量数据加工的功能,为该系统设计海量数据快速处理的API。作为数据处理平台,数据集成便是其中不可缺少的一项,本人的项目即基于此内容进行一些创新性的研发。

1.2 项目开发目的和意义

数据集成涉及将不同来源中的数据整合,并统一地向用户展示这些。数据集成在很多领域上非常重要,包括商业(当两个相似的公司需要合并它们的数据库)、科学(组合来自不同生物信息学数据库的研究结果)。 随着数据量和数据分享需求爆炸式的增长,{ 56% : 数据集成出现的频率越来越高。 }它已成为大范围理论工作的焦点,然而许多问题仍然没有解决。

上文中,维基百科的介绍已经暗示了数据集成的内容与重要性。简言之,数据集成是一个将具有不同概念、上下文、逻辑关系的数据文本进行合并,形成一个具有统一模式的数据集。作为数据分析和运用的基础,{ 57% : 计算机领域内数据集成具有重要的意义, }包括数据清洗、模式识别、生物信息等等。{ 69% : 然而,随着大数据时代的到来, }数据每天都在被生成、分析并且大量的应用着,数据驱动的决策也成为了社会中不可或缺的一部分。现如今,互联网上分散存在着海量的数据,为了充分利用这些数据中的信息以及蕴含的价值,将这些数据有效的集成在一起成为了一个显著的需求。

大数据时代下的数据集成,与传统数据集成的考核点(数量、速度、多样性、真实性)有着一定程度上的不同。首先,单一数据源无法为单一目标储存如此大量的数据。其次,很多数据源是动态的,总会有很多数据源在生成、消失,或者某些数据源改变了储存方式。第三,在诸多数据源之间,我们很难保证他们是完全统一的,这就造成了异构数据源的产生,并且相同条目下的数据也可能重复或者存在精度、修改时间等方面的差异。

为了将海量的异构、异源数据进行集成,首先需要做的就是将不同数据库的模式进行集成,生成一个全局的数据库模式,进而方便数据库记录的填充和数据库的融合。对于数据集成问题,传统的方法往往是预先指定一个全局的数据库模式,然而针对于海量数据的背景,人们难以在大量的数据中捕捉全局的信息来设计预定的全局模式,并且建立全局模式和每一个数据库模式之间的匹配关系也是耗时耗力的。

因此,在上述背景下,通过设计一些合理匹配关系和高效的算法,省时、准确的生成一个全局数据库模式成为了迫切的需求。

1.3 国内外相关领域开发及应用现状分析

自从人们开始整合有关联的数据、从中获取有用的信息开始,数据集成便走上了历史的舞台。早在计算机科学家研究这个领域之前,{ 64% : 统计学家已经做了很多的工作, }试图分析获取的数据集之间的关联。然而数据集成在很多方面都是具有挑战性的,为了解决这个问题,在过去的数十年内许多计算机科学家提出了诸如匹配连接、数据融合、数据结构化等基础性议题及可行的解决方案。近年来,无论是科学、通讯、多媒体,还是经济、医疗等方面,人们开始能够获取现实生活过程中每一件事、每一个事物的电子数据。因此,我们急切需要能够在大数据中分析、提取有效信息的能力,将现有的生活向数据驱动转变。然而最根基的一步便是数据集成。

{ 55% : 数据集成的过程大体可以分为以下三个步骤, }如图 1 1 数据集成流程所示。

图 1 1 数据集成流程

从图中我们可以看出,数据集成的基础工作就是模式集成。模式集成的目标是将不同数据库中相同或者相似的属性合并成一个属性。相同或者相似这一概念对于人们来说是可以理解的,然而对于计算机来说这个是很模糊的概念。究竟什么样的属性可以判定成相同或者相似,这是实现数据集成一个关键的问题。相关研究者大体在以下两方面进行了研究。

首先,我们需要将形式上相近的属性归为一类。在数据库的录入过程中,由于拼写错误导致的数据库质量不高的问题是很常见的。而这一问题对于数据集成来说,也是一个很显然的隐患。如何将拼写错误的属性和原属性判定为相同是一个很关键的,这里经常使用的方式是计算词间距离。我们按照一定方式计算两个属性词之间的距离,该距离在一定阈值之下,我们就可以认定两个属性是相同的。常用的方式是编辑距离[1],以一个词转变为另一个词所需的增、删、改的操作数作为两者的距离。基于此,相关的join算法已经被广泛提及[2],基于类似的算法,我们可以对多个属性集进行join操作来实现模式的集成。

除此之外,为了让计算机能够判定人类思维中相似的概念,近年来很多科学家致力于挖掘互联网上的信息并提取相关实体、关系,根据人们的思路去建立知识库,来尽可能地去表达人们对事物的认知。{ 55% : 现有的相关工作包括Freebase[3], }Google Knowledge Graph[4],ProBase[5],Yago[6]等等,这些知识库可以用来辅助我们的数据集成,使得数据集成过程能够更好地模拟人工集成的方式。如果能根据这些知识来实现模式之间

的join算法,{ 62% : 进而来实现模式集成和数据集成, }我们可以预见这一结果将是符合人们预期的。而在此角度上,{ 56% : 由于知识库是一个较新的概念, }并且知识库还不够完善,相关工作做得并不多。

基于此,我们希望能够结合形式和语义两个方面,来完成一套模式集成的系统。

1.4 模式集成相关工作介绍

作为数据库领域一项基础但非常重要的技术,模式集成已经经过了很多年的研究。过去的研究人员往往使用相似函数,诸如Jaccard相似度来处理,但是这种方法不能解决语义方面的关联。随后,论文[12]总结了模式匹配和集成领域相关的工作并进行了分类,其中大多数使用了语言学等相关知识。其中针对模式集成的工作包括DIKE[13]和ARTEMIS[14],这两种方式提出了计算属性之间相似度的方法。由于知识库是基于web中提取的数据,相比与此,我们提出的基于知识库的方法在大多情况下能够更接近于人们的做法和思路。

近些年来,微软公司也在模式集成领域进行了一些优秀的研究[15],在这篇论文中,precision和recall具有很高得结果。相比于我们基于模式(schema level)的算法,这种SEMA-JOIN算法需要更多的实例信息,即针对不同属性的值信息。对于容量较大、来源较多的异源数据库来说,将很多的值信息带入到集成过程中是不可行的。而对于生产环境中的数据库来说,不少数据库由于设计或是维护原因,部分属性下不含有值。因此,我们的算法更具有普适性。

第2章 需求分析和总体设计

为了承载所设计的模式集成算法,开发了基于知识库的海量异构数据集成系统来直观展示算法的结果并引导使用。本章主要论述了项目开发前的需求分析和设计内容。

数据集成是一个十分重要但却复杂的过程,而模式集成是其中最基础也是格外具有挑战的一个步骤。本节中将详细分析模式集成所需要进行的工作,并给出预期的设计方案。

2.1 功能概述

模式集成的主要工作是针对异源、异构数据表的模式,在形式和语义的两个维度将它们进行集成,从而得到一个统一的模式,既能将多个数据源中的所有属性全部包含,又能保证产生的数据模式中属性彼此不重复,满足数据库的范式要求。我们在此以航班数据库[7]为例,对主要工作进行分析。我们拥有5个数据库的数据,并在表 2 1 航班数据库示例中展示出它们的数据库模式。

表 2 1 航班数据库示例

数据库 表名 模式

Airline1 Schedule Flight Id, Flight Number, Start Date, End Date, Departure Time, Departure Airport, Arrival Time, Arrival Airport

Flight Flight Id, Departure Date, Departure Time, Departure Gate, Arrival Date, Arrival Time, Arrival Gate, Plane Id

Airline2 Flight Flight Number,{ 60% : Departure Airport, Scheduled Departure Date, }{ 73% : Scheduled Departure Time, Actual Departure Time, Arrival Airport, Scheduled Arrival Date, } Scheduled Arrival Time, Actual Arrival Time

Airport3 Departures Air Line, Flight Number, Scheduled, Actual, Gate Time, Takeoff Time, Terminal, Gate, Runway

Arrivals Air Line, Flight Number, Scheduled, Actual, Gate Time, Landing Time, Terminal, Gate, Runway

Airfare4 Flight Flight_Id, Flight Number, Departure Airport, Departure Date, Departure Time, Arrival Airport, Arrival Time

Fares Flight_Id, Fare Class, Fare

Airinfo5 AirportCodes Airport Code, Airport Name

AirlineCodes Air Line Code, Air Line Name

在表中我们可以发现,{ 65% : 这五个数据库之间是相互独立的, }但不同的数据表和属性之间是相等或者是重叠的。在这个例子中,模式集成的目标就是将多个数据源模式归并到一张数据表中。在这个过程中,应将含义类似(如Start Date,Takeoff Time和Scheduled Departure Date)、形式类似(如Flight_Id和Flight Id)等情况的属性合并。很显然,随着数据表数量的增大和单一数据表内属性个数的增多,逐一比较每两个属性是不合适的,我们通过实现类似于数据库范畴内的join操作,在形式和语义两个角度上完成集成。

2.2 需求分析

对于一个模式集成平台,它应能接收标准的数据库模式,并尽可能的在多个维度上完成模式集成的功能,保证前文提到的几种情况的类似属性都能够检测出并合理的整合。除了这些基本功能外,为了保证这个系统的可用性和展示效果,应设计友好的用户界面来指导完成模式集成的工作,使这个抽象的操作更容易的进行。

2.2.1 系统功能需求

本系统主要包括预处理、形式整合、语义整合、生成全局属性、前端界面展示的5个模块。本节中将以模块为单位详细介绍每个模块的功能需求。

2.2.1.1 数据预处理

数据预处理是网络训练前重要的准备工作,下面详细介绍数据预处理部分的功能需求。预处理模块主要功能点如表 2 2 预处理模块需求分析表所示。

表 2 2 预处理模块需求分析表

功能点 需求描述

模式预处理 1. 将输入的属性进行预处理,转化成词组的形式,包括以下几种情况:

ï•原词满足要求不需要改动

ï•存在缩写,根据模式说明提供的缩写规则,或是平常普遍接受的规则进行展开

ï•去除词之间的分割符号,以空格代替

ï•所有字母转换成小写

2. 为每个词组选取一个关键词代替整个词组,进而与知识库中的实体进行匹配时采用这个关键词,这里采取tf-idf算法

知识库预处理 1. 将现有的知识库进行预处理,提取实体之间的相关关系,并储存在硬盘

2. 采用合适的方式进行索引,加快磁盘中知识的访问速率,可采用的方式有:

ï•hash表

ï•B树

2.2.1.2 形式整合

形式整合模块主要功能点如表 2 3 形式整合模块需求分析表所示。

表 2 3 形式整合模块需求分析表

功能点 需求描述

生成倒排表 1. 用户给定形式上差异允许的阈值εt

2. 将用户给定的词拆分成长度为q的gram,并依据q和εt计算判定相似时所应具有相同gram的个数

3. 生成倒排表:

ï•为属性集合中的元素生成倒排表

ï•为知识库中提取出的每一个概念生成倒排表

寻找形式相近属性 1. 对于给定属性,通过倒排表筛选出形式上相近的属性

2. 将相似的所有属性构成一个集合

2.2.1.3 语义整合

语义整合模块主要功能点如表 2 4 语义整合模块需求分析表所示。

表 2 4 语义整合模块需求分析表

功能点 需求描述

寻找语义相近属性 1. 用户给定语义上差异允许的阈值γ

2. 对给定的属性在知识库范围上进行γ次join操作

后处理形式相近属性集合 1. 将判定为相似的属性按组构成集合

2. 在每组内对结果进行去重操作

3. 试图判定每个组内属性之间的语义联系,保证每个组内的属性在语义近似(γ范围内)是一个闭包

2.2.1.4 生成全局模式

生成全局属性模块主要功能点如表 2 5 生成全局属性模块需求分析表所示。

表 2 5 生成全局属性模块需求分析表

功能点 需求描述

整合形式近似和语义近似两个模块 1. 获取形式近似和语义近似两个模块的输出结果

2. 在两类结果中匹配相同属性所在的集合,并将包含同一属性的集合进行融合

3. 验证并保证新生成集合是一个形式、语义近似的闭包

生成全局模式 1. 在每个集合内,选出一个主属性,作为组内相似属性(次属性)的代表

2. 将组内其他次属性匹配至该主属性

3. 将所有输入的属性取并,并删除所有次属性

4. 建立新的全局属性到原有各个表中每个属性的关联

2.2.1.5 前端界面

前端界面模块主要功能点如表 2 6 前端界面需求分析表所示。

表 2 6 前端界面需求分析表

功能点 需求描述

输入 1. 指定输入的模式集合

2. 指定所使用的知识库

输出 1. 显示生成的全局模式

2. 显示输入模式中,每一属性与全局模式中属性的对应关系

2.2.2 系统非功能需求

本节中主要包含系统的非功能性需求,包括空间、时间限制、数据完整性等方面的需求,其具体内容如下。

1) 空间需求:本系统所储存的数据量巨大,知识库的实体数目应在百万条以上,数据库属性的个数随使用者需求而定,一般不低于一万条。程序运行过程中不应将数据成批次的读入至内存中,整个系统和相关算法应该是基于外存的。

2) 时间需求:系统运行时间应随输入模式大小而变。如遇大批量的数据进行处理时,可根据先行测试得知某一数据量处理所需大概时间,并对用户予以提示。

3) 完整性:程序运行的结果应储存在磁盘内。程序运行的过程中,应保证输入数据、所用知识库、预处理数据不被更改。

4) 扩展性:{ 56% : 系统应可根据后续需要进行扩展, }例如增加知识库处理方式、近似度判定方式等等。

5) 健壮性:系统在运行过程中,{ 62% : 应对可能出现的情况进行预测并对差错进行防范, }不会因为数据量的提升导致系统崩溃或是内存泄漏。同时前端界面应尽可能的简洁易懂,同时对用户的误操作有容忍度。

2.3 总体设计方案

2.3.1 系统功能结构

本系统根据功能划分主要包含5个模块:用户界面模块、预处理模块、形式整合模块、语义整合模块和全局模式模块。具体的功能结构图如图 2 1 系统功能结构图所示。

2.3.2 系统流程

整个系统以多个数据库的模式为输入,以现有知识库为依托,经过形式整和语义整合,生成全局属性。系统通过前端界面对以上过程的结果及必要的中间输出进行体现。该系统的流程图如图 2 2 系统流程图所示。

系统的用户提供待集成的k个模式作为系统的输入,处理过程中经历了如下的流程:

1. 形式整合:将形式上相近的属性进行匹配,将在给定阈值范围下的属性归为一组(这种情况一般发生在拼写错误的情形下)

2. 语义整合:将语义上相近的属性进行匹配,将在给定阈值范围下的属性归为一组。这一过程利用了外部的知识库(经过预处理)

3. 生成全局模式:利用前两个步骤生成的匹配属性组,在每组相似的属性中选出一个代表属性,并将其他属性在给定的k个模式中去除并对模式取并,从而得到全局模式

2.3.3 系统用例详细描述

系统用例图如图 2 3 系统用例图所示,描述了该系统实施模式集成功能时与用户的交互需求。

图 2 3 系统用例图

用例的详细描述如下:

ï•选取知识库:该系统的操作人员需要指定判断语义相似的知识库,知识库涵盖的语义范围以及精准度将直接决定系统的性能和集成效果。通常情况下待集成的模式之间应该有着必要的联系,往往是表达同一领域内的信息,因此知识库选取时更可能需要考虑对所研究范围的包含情况。{ 57%：目前完成的系统只支持Freebase, }后期可根据需要添加知识库,甚至针对某领域编写需要的相似规则

ï•给定输入模式:用户给出待集成的模式集合,为了保证集成的效果和实际意义,这些模式最好是来自接近的领域的,应有部分属性是相似的或者用来描述同一类实体的。输入的模式可能会包含很多属性,为了操作简便,我们只按照纯文本读取,暂不考虑其在数据库中的属性值以及XML等其他形式中属性之间的关系

ï•选择形近阈值:用户在执行集成前需指定 ,用以限定拼写差异的上限。现实情况中,这一环节往往是用来应对未经过较系统的数据清洗的模式,加入此模块是用来排除输入属性具有未被发现的拼写错误而产生的干扰。由于这种情况出现几率不高,为了防止对系统总体的性能和准确率产生干扰,形近阈值往往取得比较低

ï•选择语义阈值:用户在执行集成前需指定γ,用以限定语义差异的上限。这个阈值的确定很大程度上取决于输入模式和知识库的密度。知识库中相关领域的知识越密集,可能所需的阈值相对的就会大一些。阈值取的越大,进而匹配的结果包含的属性就会越多,花销时间越长,相对的全局模式可能会遗漏一些信息;而当阈值比较小时,时间开销较低,但得到的匹配结果较小,最后集成得到的全局模式越接近于输入k个模式的并集。因此这个阈值需要根据具体问题而定,目前阶段暂未有自动计算阈值的方法

ï•执行模式集成:用户给定各个配置后,启动模式集成

ï•关注集成结果:系统运行得到集成结果,用户需关注每一次得到的结果,并在小数据范围内评价该运行结果,并可能由于结果不良修改给定的阈值并重新执行

2.3.4 系统主要功能活动

2.3.4.1 模式预处理活动图

如图 2 4 模式预处理活动图所示,用户在模式预处理过程中需要和系统交替完成任务。用户打开程序,选择输入属性,系统接收属性并进行大小写、格式等归一操作,然后用户提供缩写规则,系统依据规则展开属性命中的缩写形成词组,然后用户提供分隔符,系统依据分隔符进一步分割,用户接下来选择关键词计算算法(常为tf-idf),系统根据算法计算出每个属性名中的关键词,并生成与原属性名的对应匹配规则以供下一环节使用。

图 2 4 模式预处理活动图

2.3.4.2 知识库预处理活动图

如图 2 5 知识库预处理活动图所示,用户在知识库预处理过程中需要和系统交替完成任务。用户选择使用的知识库,系统在磁盘上定位知识库信息,然后用户选择磁盘索引算法(hash、B树等),系统采用此算法将知识库中的关系散列在磁盘上,为后续操作加速。

图 2 5 知识库预处理活动图

2.3.4.3 形式整合活动图

如图 2 6 形式整合活动图所示,用户在形式整合过程中需要和系统交替完成任务。用户首先设定实验参数(形近阈值、切片大小),系统根据参数生成倒排表的形式。用户接下来一次指定输入的模式和使用的知识库,系统为这二者生成对应的倒排表。然后用户启动形式整合,系统根据倒排表和Count Filtering公式计算出形式上相近的属性/概念,并将相似的归并到统一集合中。

图 2 6 形式整合活动图

2.3.4.4 语义整合活动图

如图 2 7 语义整合活动图所示,用户在形式整合过程中需要和系统交替完成任务。用户首先设定实验参数(语义阈值),并指定输入的模式和使用的知识库,然后执行语义整合,系统根据以上设置执行多次join操作,将相似的

属性/概念放在同一个集合中,然后用户指定检验规则,系统根据规则在每个集合内去重、检验假阳性并返回结果。

图 2 7 语义整合活动图

2.3.4.5 生成全局模式活动图

如图 2 8 生成全局模式活动图所示,用户在形式整合过程中需要和系统交替完成任务。用户首先执行形近整合和语义整合,得到属性相似集合,然后用户启动全局模式生成,系统在每个相似集合内选出主属性,并在全局模式中删除主属性以外的属性进而得到全局模式。

图 2 8 生成全局模式活动图

2.3.5 开发环境和工具

2.3.5.1 开发语言

本系统采用C++语言开发并实现算法,前端展示采用web相关技术。

2.3.5.2 开发工具

如表 2 7 开发工具表所示,本系统的开发使用到了这些开发工具。

表 2 7 开发工具表

工具类别 工具名称 作用

集成开发环境 Visual Studio 2013 程序最主要的开发、调试平台

脚本开发工具 Atom 数据预处理、后处理脚本开发平台

界面设计工具 Photoshop CC2015 图片素材设计

前端开发工具 Dreamweaver 前端界面的实现

版本控制软件 Github 版本控制

2.3.5.3 开发环境

操作系统:Windows10/Ubuntu 16.04

处理器:Intel Core i7,2.40GHz主频或更高

内存:8GB 1333 MHz DDR3或更高

注:以上均为前期开发及测试时的环境,当集成到公司系统时使用了更高性能的服务器,相关运行环境发生变化,根据保密规定不予详细阐述。

第3章 核心技术及数据结构

本章节主要介绍介绍本篇论文讨论的问题,涉及的背景知识、基本技术,相关定义,以及为了使算法更为高效可靠而设计的数据结构定义。

3.1 知识库

在进行模式集成的过程中,很重要的一个问题就是如何将相似的属性归为同一条(类)属性,避免在最后生成的全局属性中产生冗余,同时又不能将不同的属性划为一类使得全局属性在信息上有遗漏。这其中一个很关键的问题就是如何判定两个属性是相似的,这个概念对于人类来说很好理解,但是交给计算机来做却是非常困难的,其中一个重要原因是人类对于判断什么样的属性是相似的具有先验知识,这个知识是在人们的成长过程中逐步积累起来、加以自己的分析和理解而形成的。而计算机没有从如此复杂背景下提取的知识和做出这样判断感性能力,同时属性是否相似还与领域、上下文息息相关,这对于计算机来说很难达成。为了解决这个问题,在我们的模式集成系统中引入了外部的知识库,尽可能通过这种方式使计算机的集成过程模拟人脑的思维。

为了判断属性名相似的情况,我们在系统中引入了知识库来进行衡量。目前阶段我们采用了Freebase 知识库,这是一个表示“is-a”关系的知识库,能够表示出相邻的两个知识概念就有类别从属或者包含的关系,相似的知识库还有WordNet , Probase 和YAGO 。这样的知识库整体的结构是一个图G,图中的每一个节点是一个概念如a和b(代表着客观存在的物体或者抽象的类别),边集S中的每条边e连接的两个节点(概念)具有从属或者包含的关系,该结构可以表示为:

虽然同义词数据库可以直接的表达我们希望得到的关系,但是现有的同义词数据库并不完善,并且在某些专业领域内,并不能简单地通过同义词来判定两个属性是相似的。而这种“is-a”关系型的知识库一定程度上可以解决我们的需求(表达数据库表之间相关联的外键结构),尤其是Freebase等知识库在专业术语范围内有着很好的

描述能力。当然采用这样形式的知识库不是唯一的,例如NGD (Normalized Google Distance)等均可以提供类似的信息。因此我们希望设计出一种统一的衡量方式和集成算法,在后期需要时可以很好地将这些知识库应用到我们的系统中。

基于上文中对语义知识库的介绍,为了将这种定性的知识间关系引入到我们的算法中,需要以一种定量的方式对此理解性的概念进行表示,这里对语义距离给出定义:

可以理解为a与b之间的距离可以表示为两点之间路径的长度(包含边的个数)。尽管这种定量表示方式可能不是非常合理,但是路径长度与语义差异程度正相关是毋庸置疑的,目前阶段取得的实验效果可以验证这种定义的定性准确性,后期的优化过程中可以试图为不同的边赋予不等的长度。基于这种定义,实际匹配的过程中需要用户指定语义近似阈值γ,当两个概念之间的距离小于γ时,可以看作这两个概念在含以上是近似的。

3.2 编辑距离与q-gram

{ 60% : 编辑距离是用来定量衡量字符串拼写差异的方法, }这种方法经过了多年的研究,例如[2][9][10]。对于 形式近似的情况,我们采用最基础的编辑距离算法,并由用户给定 作为其阈值。执行编辑距离算法时,我们为给定模式中的属性和知识库中的概念按照q-gram生成倒排表(长度q由用户给定)。具体来说,我们将词以q为单位长度进行划分,当两个词有足够多的分片相同时,他们之间的差异便会足够小,可以认为他们是相似的。我们的方法采用Count Filtering的计算方式,具体来说a与b之间如果具有 个相同的q-gram分片,我们可以认为a与b在拼写上是相近的,其中

因此,我们可以使用 作为限制条件,在字符串间使用Count Filtering进行计算得到编辑距离,并和阈值 进行比较判断是否认为相似属性。

以上的计算方法看似简单,然而由于所需比较的数据库属性和知识库中的概念数量都非常巨大,不可能在每次匹配时都将全部数据遍历一遍。为了解决这个问题,这里使用基于q-gram的倒排表索引进行解决。通过将单词切分成长度为q的分段,即单词w可以分为w1, w2, ..., wk共计 。其中wi是一个二元组(hi, vi),hi为wi的hash值,vi为一组包含wi的字符串组成。

这样,当在一组词W中查询和字符串s形式相近的词是,只需将s划分为s1, s2, ..., sk共计 个gram,并通过每个gram的hash在磁盘上反查包含它们的字符串,并统计出现的次数。当某些字符串出现次数大于 时,可以根据Count Filtering的条件判断这些字符串与s是形式上相似的。作为在整体模式集成算法中反复出现的操作,{ 55% : 通过这样基于hash的q-gram倒排表, }将逐一比对的操作转换为时间复杂度为O(1)的hash表查询,{ 59% : 大大的降低了整体算法的复杂度。 }

3.3 距离函数

基于前文给出的知识库相关内容,{ 57% : 为了定量的衡量两个属性之间的语义距离, }我们给出如下的定义:

从如上定义我们可以看出, 越小,属性 在语义层面上越相似。因此,我们可以指定一个阈值 来判定属性是否足够相似,{ 57% : 即如果两个属性之间的距离小于 , }则可以判断这两个属性是相似或相近的。

然而数据库的属性名和知识库中的概念是独立命名和维护的,所以往往两者中指向同一事物的名词在拼写上是存在差异的,甚至存在拼写录入错误。因此在衡量语义距离的同时也应考虑字面上的形近差异。我们系统中采用一个常用的字符串间距离——编辑距离[8],并以 来表示。基于以上的概念,能否判定两个属性相似的限制条件规定如下:

[Distance Constraint]

对于模式中一个属性名a,如果在知识库中存在一个概念ca,它们之间满足形近阈值 ,那么认为二者相似并可以使用ca来代替a。因此,属性a和b之间的语义距离就可以使用他们在知识库中对应概念之间的距离代替,即 等同于 。但是,当a或者b二者至少其一在知识库中找不到满足形近阈值的ca或者cb时,只能直接的用a和b之间的编辑距离来描述他们之间的差异,即 等同于 。

虽然上述的Distance Constraint在形式和语义角度都加以了限制,实际实验过程中由于语义的复杂性,还会出现假阳性的情况。例如属性import和export,在知识库中他们对应的概念(也是其本身)实际上很接近( ),但是实际上他们是一组相反的概念而并不相似。为了解决这个问题,在匹配和集成之后需要引用验证的过程,甚至包含人工操作。这些内容将在后文中详细阐述。

3.4 问题定义

模式集成的问题可以定义如下:

ïf为多个模式构成的集合,其中每个模式si可以表示为一个三元组(idi, ni, Ai),其中idi是一个唯一的id,ni是模式中属性名字的集合,Ai是这些属性对应的值的集合。

模式集成旨在生成一个全局模式Sg,包含对应的属性集合Ag和模式集合ïƒ¥ 集成的结果应该满足ïƒ中的所有模式s中的属性均能和Sg中某一属性对应,同时对应的属性之间应足够相似可以替换,即满足Distance Constraint。

该定义可以表达为:

[Schema Integration] Given a schema set ïƒ¥generate a schema Sg that satisfies that

3.5 数据结构设计

3.5.1 Cluster Set

集成系统运行过程中需要考虑语义相似和拼写相似,而其中利用知识库进行语义相似判定的部分其实是更困难的。由于知识库巨大,我们不可能将知识库完全导入到内存中,在寻找相似概念的时候也不可能遍历整个知识库。因此我们的想法是在知识库上利用类似于数据库中的join算法,只在待集成模式中的属性附近去寻找语义相似的概念。同时为了减少遍历次数,尽量使基于拼写相似的集成一同完成,我们设计了如下的数据结构Cluster Set,这种结构适用于算法执行过程中的中间结果和系统最后的输出。

[Cluster Set]

S as the concepts set of the knowledge base, a cluster set is a set of pairs {U, SU}, where U is a set of attributes and

Cluster Set实际上是一个用来储存经过匹配后的集合,其中U集合储存着被下文中算法认定为相似的属性,而为了加速这个算法在拼写相近匹配阶段的运行,SU中储存的一些属性,每个至少和U中的一个元素距离小于 。这些属性是可能在接下来的集成过程中归并到集合U中的,相比于每次在知识库上进行查找和匹配,提前将可能用到的属性以较小的代价储存下来,以空间换时间,能够提升算法的运行效率。

3.5.2 Neighbor Table

由于数据量巨大(模式中的属性和知识库中的概念目前共计在100,000,000条以上),很难将数据长期储存在内存中,因此该系统和内部的算法都是基于外存的,所以如何将数据读取的时间控制在一定范围内将是一个核心问题。通常情况下,外存的数据系统能够高效的运行取决于数据在磁盘上如何组织,hash和B树是比较常见的选择。在该数据集成系统中,为了方便高效,采用了hash的方法将知识库中的边e(a, b)储存,具体来说是分别对应起点a和b,在其hash值对应的空间内储存对应终点b和a以及其他的如边的长度等信息。

由于我们的算法实际上是在知识库中进行边的连接,形成多条路径 (start,end,length),对于不同的路径,只要它们拥有共同的终点end,便可以与边e(end, b)进行连接。因此可以将拥有共同终点的路径归为一类,多条路径只需做一次相同的扩展以提升效率,系统中使用数据结构Neighbor Table,定义如下:

[Neighbor Table]

t is an attribute and P is the set of all paths in the knowledge base. Hk(t) is a table on the disk indexed by hash value of string t, s.t.

Neighbor Table接收知识库中的一个概念t,并返回所有以t为终点、长度为k的路径,这样的结构可以以t为关键字形成一个hash表,并将生成和访问的时间保持在一个较低的数量级,边之间的连接延长可以引申为Neighbor Table的延展。类似的,边集E在这里可以表示为H1(t)。

第4章 核心算法

本章将对基于知识库的模式识别算法进行详细的介绍和分析。本系统处理的核心问题是在知识库中寻找语义和形式上相近的属性,对于模式中被判断为相似的属性归并成一条,去除冗余并得到最终的全局模式。然而知识库中所含有的概念众多,在每次搜索时将其遍历一遍是不现实的,知识库中绝大多数的信息在单次匹配中是用不上的。因此我们借鉴了数据库中的join运算符,将其思路应用到我们的系统中。面向形近相似和语义相似两种情况,提出了ED Join算法和Semantic Join算法,这两种操作都是模式集成中最基础、最重要的。

4.1 Join操作符

针对形式和语义两种情况,设计了如下的操作符:

[ED Join Operator]

Given two families of cluster sets R, T, and a threshold d, two elements (U1, S1) and (U2, S2) are from R and T, respectively are semantically joined if they satisfy one of the following constraints.

1)

2)

3)

The result of Semantic join on (U1, S1) and (U2, S2) is a pair (U, SU), where

[Semantic Join Operator]

Given two families of cluster sets R, T, and a threshold d, two elements (U1, S1) and (U2, S2) are from R and T, respectively are semantically joined if they satisfy one of the following constraints.

1)

2)

3)

The result of Semantic join on (U1, S1) and (U2, S2) is a pair (U, SU), where

为了对相似的属性进行匹配,本系统首先定义了ED Join和Semantic Join操作符 ,主要任务是将至少满足制约条件三条之一的属性认为是相似的属性,将这些属性进行归并,并根据新生成的U集的结果维护SU。首先根据第一条规则,join的操作参考r1和r2之间的相似关系,如果他们之间的距离不超过对应的阈值,他们可以认作是相似的,并将对应的集合进行join操作。如不满足第一条规则,参考之前对Cluster Set的定义,如果其中一个r1在另一个r2所拥有的S集中有对应的s,且满足r1和s之间的距离不超过 或 ,那么也可以判定相似并进行join操作。在此过程中,通过维护S集合可以避免遍历,提升匹配和join的速度。当两个集合进行join时(Pair Join),应将U集合和SU集合分别合并,并依据合并后的U来维护SU集合中的元素,此处不再详细阐述这一过程。

4.2 ED Join

ED Join用来整合具有形式上相似属性的cluster set,这里我们采用基于q-gram的编辑距离算法作为判定依据。

作为一种基本的数据结构,我们使用倒排表来描述q-gram中的每一个条目,对于输入的模式集合R和T,他们中属性的倒排表为XR(R的U集合) ,XT(T的U集合) ,ZR(R的SU集合) ,ZT(T的SU集合)。根据上节中对于判定相似的三条规则,需要在XR和XT,XR和ZT,XT和ZR之间分别比较,寻找是否存在有满足相似规则的属性对。这样的倒排表是应在预处理过程中离线生成的。

通常而言,拼写错误在知识库中是可能出现的[11],而在非集中维护的数据库的模式中更为常见。因此,降低拼写错误对集成造成的影响是引入ED Join的主要原因。该算法的伪代码如图 4 1 ED Join算法所示。

图 4 1 ED Join算法

{ 64% : 该算法大体可以分为以下几个阶段: }

1) 构造R和T的q-gram倒排表(1-4行)

2) 调用EDMerge函数进行相似属性检测(5-7行)

EDMerge函数接收倒排表H作为输入,H中每一个gram包含一个队列{v1,v2, ...},其中的每一项都含有这个gram。{ 64% : 具体流程可以分为以下几个阶段: }

1) 统计H中每个属性v的出现次数(10行)

2) 初始化结果集K(11行)

3) 对于每个出现次数超过Count Filtering中规则的属性v,比在H中能够找到和v相似的属性,这样就可以将v所在的cluster set与结果集K合并(12-16行)

该算法流程如图 4 2 ED Join流程图所示。

图 4 2 ED Join流程图

在这个过程中,时间复杂度主要取决于R和T的大小,ED Join过程中主要消耗的时间在生成q-gram这一步,{ 63% : 可以表示为O(|R|+|T|)。 }由于XR∩XT的大小一定不超过R或者T,而第10行和12-16行的时间复杂度都是O(H),即O(|XR∩XT|)不超过O(|R|+|T|)。因此,ED Join算法的时间复杂度为O(|R|+|T|)。

4.3 Semantic Join

这里的γ是用户根据数据具体情况给定的一个语义阈值,具体来说是用来限定知识库上被认定为语义相似概念的距离上限。目前阶段这个值只能通过对知识库进行分析而给出,未来希望可以通过机器学习、关联挖掘等手段进行计算。从另一个角度来说,在给定目标属性集R和知识库边集E的前提下,Semantic Join算法的功能需求是将知识库上γ距离内的属性进行归并,其过程可抽象为下面的公式:

将上述功能映射到知识库的图结构时,可以理解为是从目标属性集R起始,将边逐步进行连接成长度在 之间的路

径,这些路径的起始于R中的属性,终点属性可以看作是与R中属性相似的属性。

Semantic Join算法的具体流程如图 2 4 Semantic Join算法所示。

图 4 3 Semantic Join算法

Semantic Join算法接收的输入变量有:

ï•¬R:待集成的属性集合(以cluster set形式储存)

ï•¬γ:用户指定的语义距离阈值

ï•¬H1:知识库的边集(以neighbor table形式储存)

{ 55% : 该算法大体可以分为以下三个阶段: }

4) 初始化(1-6行):这一阶段扫描输入的属性集R,并将以R中的每一个属性r起始的路径加入到路径集合P中,其中P的定义如下:

[Path Set]

Pa is a path set, all paths in which share the same end node a, s.t.

当前过程得到的路径集合只包含了以目标属性开头的边,并按照其终点进行分组

5) 路径扩展(7-10行):这一阶段在知识库中每执行一次join操作,即对于现有得到的属性集合再向外扩展一个单位距离。整个算法共计执行 次join操作,使得能够包含与R中属性相距至多为γ的概念,即满足给定阈值范围的相似属性

6) 集合维护(11-20行):这一阶段是将相似属性所处的集合合并,即已知R1中的r1和R2中的r2相似,需要将R1和R2合并,包括合并U集合和维护对应的Su集合

该算法流程如图 4 4 Semantic Join流程图所示。

图 4 4 Semantic Join流程图

这一过程中,时间复杂度主要取决于R和P的大小。第3-5行初始化的时间复杂度为O(R),7-22是为一个范围分别为γ和|P|的二重循环,每次迭代中hash表查询操作的时间复杂度为O(1)。而执行cluster set合并的Pair Join操作的时间复杂度为O(C)(C为cluster set的平均大小),{ 74% : 所以这个循环的复杂度为O(γ|P|*|C|)。 }因此,Semantic Join算法的时间复杂度为O(|R|γ|P|*|C|)。

4.4 假阳性检验

通过对ED Join算法流程的描述可以发现,诸如“work”和“word”这种尽管没有关联但是拼写很接近的单词很可能会被认定为是相似的。而对于Semantic Join来说,如果语义阈值取得过大而知识库中概念之间的差异不小,含义相差较多的属性也可能会被判定成关联的。因此,我们通过join算法得到的结果集是很可能存在假阳性的情况,因此需要引入如下几种验证的方法。

值检验 在前文对于模式集成这一问题的描述中,我们只考虑了模式中的属性名,而没有将每个属性下的值考虑在内。值检验这一操作是通过不同属性值之间的关联和辅助判断属性之间的关系。原则上来说,如果两个属性相似到可以判定为同一条属性时,他们的值之间也应该是相同或者相似的。对于属性的值来说,结构分析尽管简单,但是往往效果都很好。这里仅简单提出几条判定规则,更详细的规则需根据实际中集成的数据库类别、来源等进行定义。

ï•¬类型:每个属性都有着数据类型,例如整数、字符串、列表、布尔值等等。拥有相同数据类型的属性是有可能相关联的,尤其是较复杂的数据结构,而如果两个属性认为相同,那么他们在非空值得数据上是极有可能具有相同的结构的。

ï•¬前后缀:前缀和后缀是英文单词中一种特殊的结构,可以辅助判断数据值之间的关联,尤其是一些特殊符号或者数据库维护人员人工定义的词组或者缩写。

人工检验 尽管值检验在某些情况中很有效,但实际上这仍是一个很弱的限制条件,对于某些数据不会生效。一般情况下,如果属性和值没有特殊的结构或者数据类型,检验过程就会很难。为了保证检验的准确度和集成的准确度,有时需要引入人工的检验和人工制定规则,众包也往往作为一种手段,这里将不再详细展开说明。

第5章 系统流程

在实际应用的过程中,往往存在着成批次大量数据集成和小量多次的增量集成,对于这两种情况,我们基础提出的算法设计了Batch Integration和Incremental Integration两种算法流程加以解决。

5.1 Batch Integration

Batch Integration是批量地集成模式中所有的属性,具体功能由ED Join和Semantic Join实现,通过这两个join操作将认为相似的属性归并到一起。Batch Integration的伪代码如图 5 1 Batch Integration伪代码所示。

图 5 1 Batch Integration伪代码

{ 60% : 在Batch Integration的流程中,所涉及的步骤如下: }

1) 将输入的模式中所有的属性都添加到集合U中(1-3行)

2) 在U集合上执行ED Join,将U集合中形近的属性归并,达到将U集合压缩减少后续处理次数的作用(4行)

3) 执行Semantic Join,将语义相似的属性进行归并(5行)

4) 在结果集U中执行分解操作,一定程度上维持属性相似的闭包,这一过程也包含假阳性的检验(6行)

该系统过程流程如图 5 2 Batch Integration流程图所示。

图 5 2 Batch Integration流程图

5.2 Incremental Integration

{ 59% : 与Batch Integration不同的是, }Incremental Integration往往基于Batch Integration得到的全局模式U,可以在较短时间内添加少量的数据,这一情形往往应用在添加数据源的操作中。Incremental Integration的基本原则是当向现有全局模式U中添加集合K中的属性时,如果K中的属性a不存在于U中时,将其添加进去并更新U对应的一系列cluster set。Incremental Integration的伪代码如图 5 2 Incremental Integration伪代码所示。

图 5 3 Incremental Integration伪代码

{ 61% : 在Incremental Integration的流程中, }所接受的输入是现有的全局模式U和新增的模式集合K,所涉及的步骤如下:

1) 将新增模式K中属性与现有全局模式U进行ED Join比对,得到相似属性所在的集合T,验证消除假阳性后得到R(1-2行)

2) 在新增集合中删除重复的属性(R集合中的属性)得到V,防止后续操作产生冗余(3行)

3) 分别将V与知识库S进行ED Join和Semantic Join,维护V中的集合,保证新增至全局模式中的信息没有缺失(4-5行)

4) 将新得到的模式信息V添加至全局模式U中,并在结果集U中执行分解操作,一定程度上维持属性相似的闭包,这一过程也包含假阳性的检验(7行)

该系统过程流程如图 5 4 Incremental Integration流程图所示。

图 5 4 Incremental Integration流程图

5.3 时间复杂度

5.3.1 Batch Integration时间复杂度

通过上文对Batch Integration流程的描述,时间复杂度如下,其中cluster set的平均长度为C。

5.3.2 Incremental Integration时间复杂度

通过上文对Incremental Integration流程的描述,时间复杂度如下:

5.3.3 时间复杂度分析

根据以上的分析,所提出的算法和流程的复杂度与知识库S的大小是无关的,算法可以在解决不同问题时基于不同大小的知识库进行集成,而无需考虑所使用的知识库的大小。时间复杂度直接收输入模式集合和输出模式集合的大小决定。

第6章 实验结果与测试

本章将对算法进行实验测试,主要指标从效率和准确性两个角度衡量。

6.1 实验设置

ï•实验环境:当前阶段采用Windows10 64-bit,Intel Core i7 2.4GHz,内存8GB

ï•数据集:{ 62% : 知识库采用Freebase, }模式来自于NYC OpenData 和SF OpenData 数据库(主要涉及范围为城市建设、基础设施等),我们从网上获取到公开的数据集并加以我们的处理

ï•阈值参数:参考2.2.3系统用例详细描述中用户对形近阈值和语义阈值的用例,这里的阈值应先期指定。根据现有的数据和知识库,此处暂且取

6.2 准确率实验

本届介绍在小数据集上对算法的精度进行的实验。由于模式集成的效果如何,直接取决于对属性匹配的结果,这里对Semantic Join算法的准确度进行了实验。

6.2.1 实验指标

为了衡量算法的效果,采用了如下的通用指标:

ï•准确率

ï•召回率

具体到实验情景而言,以上两个公式中,SA是算法找到的匹配的属性,ST是实际上相似的属性(由于相似的概念比较模糊,这里的判定参考人们的直观概念),ST∩SA是算法找出且真实相似的属性。

6.2.2 实验结果

{ 67% : 准确率结果如表 5 1 实验准确率所示。 }

表 6 1 实验准确率

Attribute Recall Precision

name 76 61 57 0.934426 0.750000

year 93 64 58 0.906250 0.617021

type 73 58 53 0.913793 0.726027

number 79 68 65 0.955882 0.822785

category 12 13 15 0.923077 0.800000

由于实验准确度的测定与人对属性之间相似度的判断有关,这里引入了人工判断作为对比组,因此这部分的实验不能有较大的数据量。对于五组中的每一组实验,都只选取了一个属性构成了待集成的模式,并验证其在使用的数据集中匹配的情况。{ 57% : 从结果中可以看到recall和precision是依据输入属性而变化的, }recall的平均值为0.9266862,precision的平均值为0.7431666。

6.2.3 实验分析

如上的实验是以人的判断作为绝对准则,人是通过多年积累的背景知识进行判断的。Semantic Join算法作为人工判断的一种模拟手段,由于不具有较广的背景知识、较灵活的知识应用,固然和人工匹配有一定差距,但是在某些程度上还是有着不错的表现,具体分析和改进如下:

ï•关于recall的分析:recall这一指标大体的反映了算法结果与人工匹配的吻合程度,简言之,人们判定相似的属性有多少被算法识别了出来。实验结果中recall大体在0.9左右浮动,{ 60% : 可以看出算法的识别率比较高, }不能识别的很多是因为知识库和模式中的词语不能完全对应,这一点将在2.3存在的困难与问题进行讨论

{ 55% : ï•关于precision的分析: }precision这一指标大体的反映了算法结果的准确度,简言之,算法认为匹配的属性有多少是真正相似的。相比于recall,实验结果中precision的值是相对较低的。有些不准确的可能是知识库分布不均匀导致的,这一点将在2.3存在的困难与问题进行讨论

6.3 效率实验

对于面向于大规模数据的算法,其效率是十分关键的,因此我们在此针对效率进行了实验。对于我们提出的算法来说,其运行时间受到多方面因素的影响,包括数据量、进行的属性、形近阈值和语义阈值大小等等。在本节的实验中,知识库采用来自于Freebase中9,471,476条真实知识库数据。

6.3.1 数据量的影响

为了考察数据量大小对于时间消耗的影响,我们为多组实验的输入设置了不同的数据量,结果结合的大小也随之改变。部分实验结果如表 6 2 运行时间受数据量的影响(batch integration)所示。直观上看,运行时间随着数据量的增长和增加。

表 6 2 运行时间受数据量的影响(batch integration)

Input set No. Input size Result size Running time

1 1 47 0.015

2 1 267 0.021

3 1 47467 0.672

4 5 12073 0.453

5 5 201529 5.625

6 10 106 0.084

7 10 19207 0.582

8 20 252163 51.13

9 20 84962 2.573

10 30 99243 12.39

11 30 177027 20.979

12 40 188034 30.185

13 40 376257 66.327

14 50 189247 18.009

15 50 204929 30.384

具体而言,数据量大小可以分为输入数据集大小和输出数据集大小。从表 6 2 运行时间受数据量的影响(batch integration)我们可以看出,运行时间更受到输出数据量大小的影响。究其原因,在知识库上不同的概念之间周围疏密度是不同的,如果集成过程中包含有着很多邻居的概念(处于图的稠密部分),往往处理过程是更耗时间的。

6.3.2 阈值的影响

为了考察数据量大小对于时间消耗的影响,我们为多组实验的输入设置了不同的语义阈值。为了放置的道德结果数目过小、实现效果不明显,我们将阈值的下限设为2。实验结果如表 6 3 运行时间受阈值的影响(batch integration)和图 6 1 运行时间受阈值的影响(batch integration)所示。

表 6 3 运行时间受阈值的影响(batch integration)

Input size Threshold Result size Running time

1 2 66 0.024

1 3 213 0.129

1 4 69892 36.641

2 2 89 0.039

2 3 4145 0.304

2 4 108493 101.54

3 2 730 0.103

3 3 7086 3.481

3 4 111161 131.434

图 6 1 运行时间受阈值的影响(batch integration)

{ 62% : 对于batch integration, }从结果中我们可以发现阈值对于时间消耗的影响很大。当阈值超过一定值是,随着阈值的增加,运行时间显著的提升。究其原因,是因为当阈值超过一定范围是,知识库中会被集成的概念越来越多,而且增长率远超线性。

{ 59% : 因此,在系统的实际使用过程中, }应事先考虑到目标模式、知识库、实际需要的相似度等来确定阈值的取值。同时,这个阈值也不应设置的过高,否则会导致预期的相似度下降,差别较大的属性被认为相似,进而导致最终得到的全局属性存在数据的缺失。

6.3.3 Cluster set结构的影响

为了增加incremental integration过程中的效率,我们设计了cluster set这种结构,预先储存部分信息来加速后

续处理。本节中将通过实验来验证这种数据结构存在的必要性,实验结果如图 6 2 运行时间受cluster set的影响(incremental integration)所示。

图 6 2 运行时间受cluster set的影响(incremental integration)

根据图中结果,我们可以看出当添加的属性在现有集合中存在时,其判断和处理所需的运行时间要相对较低。这样对于出现多次的属性来说可以节省很多时间,而在实际应用场景中,模式集成往往是在相同或者相近领域的数据库间进行的,这种同一属性多次出现是很常见的。

结 论

在毕业设计中,我们提出了一种基于海量异构数据的模式集成算法,在本文中将算法以及相关的结果、系统架构进行了详细的阐述,基于不同的应用场景提出了适用于大批量模式进行集成的batch integration和小范围增量集成的incremental integration。为了达到更好的集成效果,我们引入了形式和语义近似的判断条件,形式近似采用了基于q-gram的编辑距离算法进行衡量,对于含义相近的属性采用近年来日趋完善的知识库进行判断。实验结果表明,我们提出的算法能够高效、准确的解决模式集成的任务。

考虑到当前阶段,{ 59%：现有的知识库所能提供的信息有限, }不能如同期待中一样很准确的衡量属性之间的语义关联,接线来的工作可以通过撰写规则等方式,在现有的知识库中进行数据挖掘,为概念之间的关联度进行定量的赋值,而不是简单的使用在知识图谱中的距离表示予以差异。相信通过这些的工作,可能提升模式集成的准确度。