Sven Hartmann · Hui Ma
Abdelkader Hameurlain
Günther Pernul
Roland R. Wagner (Eds.)

LNCS 11030

Database and Expert Systems Applications

29th International Conference, DEXA 2018
Regensburg, Germany, September 3–6, 2018
Proceedings, Part II

2 Part II

DEXA 2018

Springer

Lecture Notes in Computer Science 11030

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

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

Sven Hartmann · Hui Ma
Abdelkader Hameurlain · Günther Pernul
Roland R. Wagner (Eds.)

Database and Expert Systems Applications

29th International Conference, DEXA 2018
Regensburg, Germany, September 3–6, 2018
Proceedings, Part II

Springer

Editors
Sven Hartmann
Clausthal University of Technology
Clausthal-Zellerfeld
Germany

Hui Ma
Victoria University of Wellington
Wellington
New Zealand

Abdelkader Hameurlain
Paul Sabatier University
Toulouse
France

Günther Pernul
University of Regensburg
Regensburg
Germany

Roland R. Wagner
Johannes Kepler University
Linz
Austria

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-98811-5 ISBN 978-3-319-98812-2 (eBook)
https://doi.org/10.1007/978-3-319-98812-2

Library of Congress Control Number: 2018950775

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

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

This volume contains the papers presented at the 29th International Conference on Database and Expert Systems Applications (DEXA 2018), which was held in Regensburg, Germany, during September 3–6, 2018. On behalf of the Program Committee, we commend these papers to you and hope you find them useful.

Database, information, and knowledge systems have always been a core subject of computer science. The ever-increasing need to distribute, exchange, and integrate data, information, and knowledge has added further importance to this subject. Advances in the field will help facilitate new avenues of communication, to proliferate interdisciplinary discovery, and to drive innovation and commercial opportunity.

DEXA is an international conference series that showcases state-of-the-art research activities in database, information, and knowledge systems. The conference and its associated workshops provide a premier annual forum to present original research results and to examine advanced applications in the field. The goal is to bring together developers, scientists, and users to extensively discuss requirements, challenges, and solutions in database, information, and knowledge systems.

DEXA 2018 solicited original contributions dealing with any aspect of database, information, and knowledge systems. Suggested topics included, but were not limited to:

- Acquisition, Modeling, Management, and Processing of Knowledge
- Authenticity, Privacy, Security, and Trust
- Availability, Reliability, and Fault Tolerance
- Big Data Management and Analytics
- Consistency, Integrity, Quality of Data
- Constraint Modeling and Processing
- Cloud Computing and Database-as-a-Service
- Database Federation and Integration, Interoperability, Multi-Databases
- Data and Information Networks
- Data and Information Semantics
- Data Integration, Metadata Management, and Interoperability
- Data Structures and Data Management Algorithms
- Database and Information System Architecture and Performance
- Data Streams and Sensor Data
- Data Warehousing
- Decision Support Systems and Their Applications
- Dependability, Reliability, and Fault Tolerance
- Digital Libraries and Multimedia Databases
- Distributed, Parallel, P2P, Grid, and Cloud Databases
- Graph Databases
- Incomplete and Uncertain Data
- Information Retrieval

- Information and Database Systems and Their Applications
- Mobile, Pervasive, and Ubiquitous Data
- Modeling, Automation, and Optimization of Processes
- NoSQL and NewSQL Databases
- Object, Object-Relational, and Deductive Databases
- Provenance of Data and Information
- Semantic Web and Ontologies
- Social Networks, Social Web, Graph, and Personal Information Management
- Statistical and Scientific Databases
- Temporal, Spatial, and High-Dimensional Databases
- Query Processing and Transaction Management
- User Interfaces to Databases and Information Systems
- Visual Data Analytics, Data Mining, and Knowledge Discovery
- WWW and Databases, Web Services
- Workflow Management and Databases
- XML and Semi-structured Data

Following the call for papers, which yielded 160 submissions, there was a rigorous review process that saw each submission refereed by three to six international experts. The 35 submissions judged best by the Program Committee were accepted as full research papers, yielding an acceptance rate of 22%. A further 40 submissions were accepted as short research papers.

As is the tradition of DEXA, all accepted papers are published by Springer. Authors of selected papers presented at the conference were invited to submit substantially extended versions of their conference papers for publication in the Springer journal *Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS)*. The submitted extended versions underwent a further review process.

The success of DEXA 2018 was the result of collegial teamwork from many individuals. We wish to thank all authors who submitted papers and all conference participants for the fruitful discussions.

We are grateful to Xiaofang Zhou (The University of Queensland) for his keynote talk on "Spatial Trajectory Analytics: Past, Present, and Future" and to Tok Wang Ling (National University of Singapore) for his keynote talk on "Data Models Revisited: Improving the Quality of Database Schema Design, Integration and Keyword Search with ORA-Semantics."

This edition of DEXA also featured three international workshops covering a variety of specialized topics:

- BDMICS 2018: Third International Workshop on Big Data Management in Cloud Systems
- BIOKDD 2018: 9th International Workshop on Biological Knowledge Discovery from Data
- TIR 2018: 15th International Workshop on Technologies for Information Retrieval

We would like to thank the members of the Program Committee and the external reviewers for their timely expertise in carefully reviewing the submissions. We are grateful to our general chairs, Abdelkader Hameurlain, Günther Pernul, and

Roland R. Wagner, to our publication chair, Vladimir Marik, and to our workshop chairs, A Min Tjoa and Roland R. Wagner.

We wish to express our deep appreciation to Gabriela Wagner of the DEXA conference organization office. Without her outstanding work and excellent support, this volume would not have seen the light of day.

Finally, we like to thank Günther Pernul and his team for being our hosts during the wonderful days in Regensburg.

July 2018

Sven Hartmann
Hui Ma

Organization

General Chairs

Abdelkader Hameurlain	IRIT, Paul Sabatier University, Toulouse, France
Günther Pernul	University of Regensburg, Germany
Roland R. Wagner	Johannes Kepler University Linz, Austria

Program Committee Chairs

Hui Ma	Victoria University of Wellington, New Zealand
Sven Hartmann	Clausthal University of Technology, Germany

Publication Chair

Vladimir Marik	Czech Technical University, Czech Republic

Program Committee

Slim Abdennadher	German University, Cairo, Egypt
Hamideh Afsarmanesh	University of Amsterdam, The Netherlands
Riccardo Albertoni	Institute of Applied Mathematics and Information Technologies - Italian National Council of Research, Italy
Idir Amine Amarouche	University Houari Boumediene, Algeria
Rachid Anane	Coventry University, UK
Annalisa Appice	Università degli Studi di Bari, Italy
Mustafa Atay	Winston-Salem State University, USA
Faten Atigui	CNAM, France
Spiridon Bakiras	Hamad bin Khalifa University, Qatar
Zhifeng Bao	National University of Singapore, Singapore
Ladjel Bellatreche	ENSMA, France
Nadia Bennani	INSA Lyon, France
Karim Benouaret	Université Claude Bernard Lyon 1, France
Benslimane Djamal	Lyon 1 University, France
Morad Benyoucef	University of Ottawa, Canada
Catherine Berrut	Grenoble University, France
Athman Bouguettaya	University of Sydney, Australia
Omar Boussaid	University of Lyon/Lyon 2, France
Stephane Bressan	National University of Singapore, Singapore
Barbara Catania	DISI, University of Genoa, Italy
Michelangelo Ceci	University of Bari, Italy
Richard Chbeir	UPPA University, France

Cindy Chen	University of Massachusetts Lowell, USA
Phoebe Chen	La Trobe University, Australia
Max Chevalier	IRIT - SIG, Université de Toulouse, France
Byron Choi	Hong Kong Baptist University, Hong Kong, SAR China
Soon Ae Chun	City University of New York, USA
Deborah Dahl	Conversational Technologies, USA
Jérôme Darmont	Université de Lyon (ERIC Lyon 2), France
Roberto De Virgilio	Università Roma Tre, Italy
Vincenzo Deufemia	Università degli Studi di Salerno, Italy
Gayo Diallo	Bordeaux University, France
Juliette Dibie-Barthélemy	AgroParisTech, France
Dejing Dou	University of Oregon, USA
Cedric du Mouza	CNAM, France
Johann Eder	University of Klagenfurt, Austria
Suzanne Embury	The University of Manchester, UK
Markus Endres	University of Augsburg, Germany
Noura Faci	Lyon 1 University, France
Bettina Fazzinga	ICAR-CNR, Italy
Leonidas Fegaras	The University of Texas at Arlington, USA
Stefano Ferilli	University of Bari, Italy
Flavio Ferrarotti	Software Competence Center Hagenberg, Austria
Vladimir Fomichov	School of Business Informatics, National Research University Higher School of Economics, Moscow, Russian Federation
Flavius Frasincar	Erasmus University Rotterdam, The Netherlands
Bernhard Freudenthaler	Software Competence Center Hagenberg, Austria
Hiroaki Fukuda	Shibaura Institute of Technology, Japan
Steven Furnell	Plymouth University, UK
Joy Garfield	University of Worcester, UK
Claudio Gennaro	ISTI-CNR, Italy
Manolis Gergatsoulis	Ionian University, Greece
Javad Ghofrani	Leibniz Universität Hannover, Germany
Fabio Grandi	University of Bologna, Italy
Carmine Gravino	University of Salerno, Italy
Sven Groppe	Lübeck University, Germany
Jerzy Grzymala-Busse	University of Kansas, USA
Francesco Guerra	Università degli Studi di Modena e Reggio Emilia, Italy
Giovanna Guerrini	University of Genoa, Italy
Allel Hadjali	ENSMA, Poitiers, France
Abdelkader Hameurlain	Paul Sabatier University, France
Ibrahim Hamidah	Universiti Putra Malaysia, Malaysia
Takahiro Hara	Osaka University, Japan
Sven Hartmann	Clausthal University of Technology, Germany
Wynne Hsu	National University of Singapore, Singapore

Yu Hua	Huazhong University of Science and Technology, China
San-Yih Hwang	National Sun Yat-Sen University, Taiwan
Theo Härder	TU Kaiserslautern, Germany
Ionut Emil Iacob	Georgia Southern University, USA
Sergio Ilarri	University of Zaragoza, Spain
Abdessamad Imine	Inria Grand Nancy, France
Yasunori Ishihara	Nanzan University, Japan
Peiquan Jin	University of Science and Technology of China, China
Anne Kao	Boeing, USA
Dimitris Karagiannis	University of Vienna, Austria
Stefan Katzenbeisser	Technische Universität Darmstadt, Germany
Anne Kayem	Hasso Plattner Institute, Germany
Carsten Kleiner	University of Applied Sciences and Arts Hannover, Germany
Henning Koehler	Massey University, New Zealand
Harald Kosch	University of Passau, Germany
Michal Krátký	Technical University of Ostrava, Czech Republic
Petr Kremen	Czech Technical University in Prague, Czech Republic
Sachin Kulkarni	Macquarie Global Services, USA
Josef Küng	University of Linz, Austria
Gianfranco Lamperti	University of Brescia, Italy
Anne Laurent	LIRMM, University of Montpellier 2, France
Lenka Lhotska	Czech Technical University, Czech Republic
Yuchen Li	Singapore Management University, Singapore
Wenxin Liang	Dalian University of Technology, China
Tok Wang Ling	National University of Singapore, Singapore
Sebastian Link	The University of Auckland, New Zealand
Chuan-Ming Liu	National Taipei University of Technology, Taiwan
Hong-Cheu Liu	University of South Australia, Australia
Jorge Lloret Gazo	University of Zaragoza, Spain
Alessandra Lumini	University of Bologna, Italy
Hui Ma	Victoria University of Wellington, New Zealand
Qiang Ma	Kyoto University, Japan
Stephane Maag	TELECOM SudParis, France
Zakaria Maamar	Zayed University, United Arab Emirates
Elio Masciari	ICAR-CNR, Università della Calabria, Italy
Brahim Medjahed	University of Michigan - Dearborn, USA
Harekrishna Mishra	Institute of Rural Management Anand, India
Lars Moench	University of Hagen, Germany
Riad Mokadem	IRIT, Paul Sabatier University, France
Yang-Sae Moon	Kangwon National University, South Korea
Franck Morvan	IRIT, Paul Sabatier University, France
Dariusz Mrozek	Silesian University of Technology, Poland
Francesc Munoz-Escoi	Universitat Politecnica de Valencia, Spain
Ismael Navas-Delgado	University of Málaga, Spain

Wilfred Ng	Hong Kong University of Science and Technology, Hong Kong, SAR China
Javier Nieves Acedo	IK4-Azterlan, Spain
Mourad Oussalah	University of Nantes, France
George Pallis	University of Cyprus, Cyprus
Ingrid Pappel	Tallinn University of Technology, Estonia
Marcin Paprzycki	Polish Academy of Sciences, Warsaw Management Academy, Poland
Oscar Pastor Lopez	Universitat Politecnica de Valencia, Spain
Francesco Piccialli	University of Naples Federico II, Italy
Clara Pizzuti	Institute for High Performance Computing and Networking (ICAR)-National Research Council (CNR), Italy
Pascal Poncelet	LIRMM, France
Elaheh Pourabbas	National Research Council, Italy
Claudia Raibulet	Università degli Studi di Milano-Bicocca, Italy
Praveen Rao	University of Missouri-Kansas City, USA
Rodolfo Resende	Federal University of Minas Gerais, Brazil
Claudia Roncancio	Grenoble University/LIG, France
Massimo Ruffolo	ICAR-CNR, Italy
Simonas Saltenis	Aalborg University, Denmark
N. L. Sarda	I.I.T. Bombay, India
Marinette Savonnet	University of Burgundy, France
Florence Sedes	IRIT, Paul Sabatier University, Toulouse, France
Nazha Selmaoui	University of New Caledonia, New Caledonia
Michael Sheng	Macquarie University, Australia
Patrick Siarry	Université Paris 12 (LiSSi), France
Gheorghe Cosmin Silaghi	Babes-Bolyai University of Cluj-Napoca, Romania
Hala Skaf-Molli	Nantes University, France
Bala Srinivasan	Retried, Monash University, Australia
Umberto Straccia	ISTI - CNR, Italy
Maguelonne Teisseire	Irstea - TETIS, France
Sergio Tessaris	Free University of Bozen-Bolzano, Italy
Olivier Teste	IRIT, University of Toulouse, France
Stephanie Teufel	University of Fribourg, Switzerland
Jukka Teuhola	University of Turku, Finland
Jean-Marc Thevenin	University of Toulouse 1 Capitole, France
A Min Tjoa	Vienna University of Technology, Austria
Vicenc Torra	University of Skövde, Sweden
Traian Marius Truta	Northern Kentucky University, USA
Theodoros Tzouramanis	University of the Aegean, Greece
Lucia Vaira	University of Salento, Italy
Ismini Vasileiou	University of Plymouth, UK
Krishnamurthy Vidyasankar	Memorial University of Newfoundland, Canada
Marco Vieira	University of Coimbra, Portugal
Junhu Wang	Griffith University, Brisbane, Australia

Wendy Hui Wang	Stevens Institute of Technology, USA
Piotr Wisniewski	Nicolaus Copernicus University, Poland
Ming Hour Yang	Chung Yuan Christian University, Taiwan
Yang, Xiaochun	Northeastern University, China
Yanchang Zhao	CSIRO, Australia
Qiang Zhu	The University of Michigan, USA
Marcin Zimniak	Leipzig University, Germany
Ester Zumpano	University of Calabria, Italy

Additional Reviewers

Valentyna Tsap	Tallinn University of Technology, Estonia
Liliana Ibanescu	AgroParisTech, France
Cyril Labbé	Université Grenoble-Alpes, France
Zouhaier Brahmia	University of Sfax, Tunisia
Dunren Che	Southern Illinois University, USA
Feng George Yu	Youngstown State University, USA
Gang Qian	University of Central Oklahoma, USA
Lubomir Stanchev	Cal Poly, USA
Jorge Martinez-Gil	Software Competence Center Hagenberg, Austria
Loredana Caruccio	University of Salerno, Italy
Valentina Indelli Pisano	University of Salerno, Italy
Jorge Bernardino	Polytechnic Institute of Coimbra, Portugal
Bruno Cabral	University of Coimbra, Portugal
Paulo Nunes	Polytechnic Institute of Guarda, Portugal
William Ferng	Boeing, USA
Amin Mesmoudi	LIAS/University of Poitiers, France
Sabeur Aridhi	LORIA, University of Lorraine - TELECOM Nancy, France
Julius Köpke	Alpen Adria Universität Klagenfurt, Austria
Marco Franceschetti	Alpen Adria Universität Klagenfurt, Austria
Meriem Laifa	Bordj-Bouarreridj University, Algeria
Sheik Mohammad Mostakim Fattah	University of Sydney, Australia
Mohammed Nasser Mohammed Ba-hutair	University of Sydney, Australia
Ali Hamdi Fergani Ali	University of Sydney, Australia
Masoud Salehpour	University of Sydney, Australia
Adnan Mahmood	Macquarie University, Australia
Wei Emma Zhang	Macquarie University, Australia
Zawar Hussain	Macquarie University, Australia
Hui Luo	RMIT University, Australia
Sheng Wang	RMIT University, Australia
Lucile Sautot	AgroParisTech, France
Jacques Fize	Cirad, Irstea, France

María del Carmen Rodríguez-Hernández	Technological Institute of Aragón, Spain
Ramón Hermoso	University of Zaragoza, Spain
Senen Gonzalez	Software Competence Center Hagenberg, Austria
Ermelinda Oro	High Performance and Computing Institute of the National Research Council (ICAR-CNR), Italy
Shaoyi Yin	Paul Sabatier University, France
Jannai Tokotoko	ISEA University of New Caledonia, New Caledonia
Xiaotian Hao	Hong Kong University of Science and Technology, Hong Kong, SAR China
Ji Cheng	Hong Kong University of Science & Technology, Hong Kong, China
Radim Bača	Technical University of Ostrava, Czech Republic
Petr Lukáš	Technical University of Ostrava, Czech Republic
Peter Chovanec	Technical University of Ostrava, Czech Republic
Galicia Auyon Jorge Armando	ISAE-ENSMA, Poitiers, France
Nabila Berkani	ESI, Algiers, Algeria
Amine Roukh	Mostaganem University, Algeria
Chourouk Belheouane	USTHB, Algiers, Algeria
Angelo Impedovo	University of Bari, Italy
Emanuele Pio Barracchia	University of Bari, Italy
Arpita Chatterjee	Georgia Southern University, USA
Stephen Carden	Georgia Southern University, USA
Tharanga Wickramarachchi	U.S. Bank, USA
Divine Wanduku	Georgia Southern University, USA
Lama Saeeda	Czech Technical University in Prague, Czech Republic
Michal Med	Czech Technical University in Prague, Czech Republic
Franck Ravat	Université Toulouse 1 Capitole - IRIT, France
Julien Aligon	Université Toulouse 1 Capitole - IRIT, France
Matthew Damigos	Ionian University, Greece
Eleftherios Kalogeros	Ionian University, Greece
Srini Bhagavan	IBM, USA
Monica Senapati	University of Missouri-Kansas City, USA
Khulud Alsultan	University of Missouri-Kansas City, USA
Anas Katib	University of Missouri-Kansas City, USA
Jose Alvarez	Telecom SudParis, France
Sarah Dahab	Telecom SudParis, France
Dietrich Steinmetz	Clausthal University of Technology, Germany

Contents – Part II

Data Streams

Information Networks and Algorithms

Database System Architecture and Performance

Learning

Emerging Applications

Data Mining

Contents – Part I

Data Semantics

Cloud Data Processing

Time Series Data

Social Networks

Information Retrieval

Template Trees: Extracting Actionable Information from Machine Generated Emails

Manoj K. Agarwal(✉) and Jitendra Singh

AI & Research, Microsoft – India, Hyderabad 500032, India
{agarwalm, jisingh}@microsoft.com

Abstract. Many machine generated emails carry important information which must be acted upon at scheduled time by the recipient. Thus, it becomes a natural goal to automatically extract such actionable information from these emails and communicate to the users. These emails are generated for many different domains, providing different types of services. However, such emails carry personal information, therefore, it becomes difficult to get access to large corpus of labeled data for supervised information extraction methods.

In this paper, we propose a novel method to automatically identify part of the email containing actionable information, called core region of the email, with the aid of a domain dictionary. Domain dictionary is generated based on the public information of the domain. The core regions are stored as template trees - a template tree is a sub-tree embedded in the email's HTML DOM tree.

Our experiments over real data show, structure of the core region of the email, containing all the information of our interest, is very simple and it is 85%–98% smaller compared to the original email. Further, our experiments also show that the template trees are highly repetitive across diverse set of emails from a given service provider.

Keywords: Email · Templates · Trees · Information retrieval
Algorithm

1 Introduction

In a recent study, it is showed that 90% of Web mail traffic is machine generated [1, 7, 8, 10]. As mentioned in [1], "*A common characteristics of these machine generated messages is that most of them are highly structured documents, with rich HTML formatting, and they are repeated over and over, modulo minor variations, in the global mail corpus. These characteristics clearly facilitate the application of automated data extraction and learning methods at a very large scale*". With a significant chunk of web mail traffic composed of machine generated emails, it becomes a natural goal to automatically extract the information from these emails, which must be acted upon by the recipient by scheduled time, e.g., payment of utility bill by the due date. Therefore, we define actionable information as "*information that must be acted upon by scheduled time, by the recipient*". The requirement to extract actionable information from machine generated emails is present for many domains, such as flight information, shipment arrival, payment due date, etc., [5].

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 3–18, 2018.
https://doi.org/10.1007/978-3-319-98812-2_1

A big challenge faced by such systems is: *such emails contain personal information*. Therefore, it becomes difficult to get access to large corpus of labeled data to build annotators based on supervised methods [1, 3]. Moreover, even though machine generated emails are structured, their structure changes over time. Further, new service providers join continuously. Hence, it becomes difficult for supervised methods to model the tail traffic. For example, hotel reservation emails have more than 8000 different small providers representing more than 50% of the total such emails in our database. Similarly, we have over one thousand small service providers for flight data, representing more than 40% of all flight reservation emails. There exist supervised methods [8, 9] that work on labeled data with varying degree of success.

In this paper, we present a novel method to extract actionable information from emails. We show that the region in the email containing actionable information can be represented as combination of one or more small template trees. These template trees have significantly simpler structure compared to the original email. Further, these template trees are highly repetitive across a diverse set of email from a given service provider and contain all the information of our interest. We develop a principled and scalable approach which exploits the semi-structured format of the emails to extract these template trees. Requirement of training data, for building supervised annotators over email data, can be reduced significantly, if such small and well-structured snippets from the emails can be identified for information extraction.

Only information needed by our system is a domain specific dictionary. We call it **domain knowledge**. The domain knowledge has been used in earlier systems to automatically extract the information from HTML web pages [15]. Similarly, there are existing systems to extract wrappers from HTML pages [16]. However, unlike a wrapper a template tree is a sub-tree in the email HTML DOM tree. In [17], authors present a system to extract templates of an entire web pages in an unsupervised manner. On the other hand, we templatize only the structure of the core region of the emails, containing the information of our interest, with the aid of domain knowledge.

However, the work in [16, 17] exposes that machine generated HTML documents (web pages or emails) have a fixed structure, encoded with the actual data. We present a novel system, that builds on these ideas to extract actionable information from emails.

We use the term ‘domain’ for an entire service. For instance, for extracting information from flight related emails, ‘flight’ is the domain. A specific vendor in a domain is called service provider, or just ‘provider’. In ‘flight’ domain, each airline is a provider (e.g., ‘American Airlines’). We will use the emails from flight domain as running example throughout the paper.

Domain specific dictionaries contain keywords and regex patterns specific to that domain. Dictionaries are applicable on entire domain and are not specific to any provider. Therefore, they are built using public information and it is much easier to acquire and learn domain specific dictionaries as opposed to acquiring labelled data for every provider (cf. Sect. 3.2).

The technique described in this paper is applicable across many domains for which domain specific information can be represented in a dictionary. This is a highly generic requirement, applicable across most domains, producing machine generated emails.

1.1 Contributions

- We present a novel system to extract actionable information from machine generated emails. Our system needs just the domain knowledge in a dictionary for the entire domain. Our system automatically learns new templates.
- We introduce a novel mechanism, using DeweyIds [13], which helps us maintain and update the database of template trees highly efficiently.
- We present our results on real email data. Our results show that template trees are much simpler and smaller in structure; and our system can model a diverse set of emails, from a service provider using a small number of templates.

The organization of the paper is as follows: In Sect. 2 we present the related work. In Sect. 3, we present our methodology to construct domain specific dictionaries. In Sect. 4, we describe our technique to identify core region in the email. We present our methodology to identify sub-templates in Sect. 5. In Sect. 6 we present our results followed by conclusion in Sect. 7.

2 Related Work

One of the first methods to learn templates from email was reported in [11]. They learn templates from the email subject, over a representative sample of emails. In [3], the authors present a statistical method to extract product names from email. Their system introduces the concept of email templates and domain specific dictionaries, but the focus of work in [3] is to annotate product names from the extracted text. In [8], the authors propose a method to discover templates in the email but here template implies a fixed phrase repeating across emails, whereas, for our problem presented in this paper, a template implies a fixed DOM structure along with phrases.

In [4], authors present a method to mine 'Data Records' in a Web Page. Although the technique is not tailored for emails, they extract 'Data Record' as a core region from an HTML web page. The concept of 'core region' is similar to our concept core-region. However, they use an edit-distance based method to identify the core region and makes specific assumptions about the email structure. Even a minor variation in the email structure will make the method un-scalable.

In [2, 6], authors present a method to categorize incoming emails into categories such as bills, itineraries, promotions, receipts, etc. In [2], the authors propose a text based template generation method, as opposed to tree-based templates in our system. In [6], the objective is to assign one of the predefined categories to an incoming email. In [7], the objective is to predict the distribution of the category of mail that may arrive over a given time window.

In [10], authors exploit the DOM tree of email HTML structure to cluster emails in one of a prespecified m clusters. The idea to exploit HTML DOM structure to cluster similar emails was first explored in [1]. However, template trees are different from just matching the DOM structure (Sect. 4.1).

3 Preliminaries

As reported in [1], machine generated emails are HTML formatted structured documents, modulo minor variations. We exploit this observation in our methodology.

3.1 Tree Representation of the Email

All machine generated emails, produced by a provider and providing similar information to its recipients, have same HTML structure modulo variations in the text of the text nodes. We exploit this property to cluster similar emails. We identify the sub-trees embedded in the DOM structure of email HTML tree that contain the information of our interest as templates. We use the DOM tree structure to encode the templates because (1) the text node of a template tree gives us well defined start and end positions of the core region; (2) it is highly unlikely that two emails generated by two different providers will have similar DOM structure [3]; (3) if the DOM structure of two emails match, with a high probability, these emails are generated by same provider and have similar information. Hence, DOM tree based templates gives us a high precision in clustering similar emails.

Email Parsing: While parsing the email, we get rid of HTML styling information. Thus, we discard the attributes of HTML tags. We also get rid of all HTML tag (<P>, <H>, etc.). Instead, we assign a DeweyId [13] to each node in the DOM tree. A DeweyId exactly describes the location of a node in the tree. A node with DeweyId 0.1.1 is the 2nd child of its parent node 0.1. Thus, an HTML email, as shown in Fig. 1, is converted into a tree shown in Fig. 2.

```
<html> <body> <table><tbody>
    <tr> <td>
    <span>Flight</span> <strong>San Francisco, CA</strong> (SFO) to
<strong>Seattle, WA (SEA)</strong>
    </td> </tr>
    <tr> <td> Alaska Airlines <strong>123</strong>
     </td></tr>
</tbody></table></body></html>
```

Fig. 1. Example HTML tree from a flight email

There are two types of nodes in this tree: *Internal nodes*, which are non-text nodes and *leaf nodes* which are text nodes. We keep a hash table with node's DeweyId as the key. For internal nodes, it points to its children and its parent node. For leaf nodes it points to its text value and parent node. Tree and hash tables are prepared in a single pass, while parsing the email.

Our objective is to identify the smallest sub-tree in the email tree that contains all the text nodes of our interest. A text node is of our interest if it contains a token from the Domain dictionary. This sub-tree is called the core region and is denoted by *iTree* (short for information tree). In next section, we show how we prepare the domain specific dictionary.

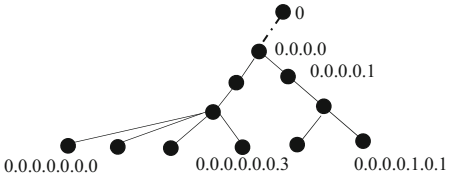


Fig. 2. The structure of the DOM tree of HTML in Fig. 1, encoded with the DeweyIds

3.2 Domain Dictionary

A domain dictionary contains the most relevant keywords along with relevant regex patterns, to identify commonly occurring keywords and patterns specific to the domain. We use 'token' to refer a keyword or a regex pattern in the domain dictionary. Since we use 'flight' domain as our running example, below we describe the method to prepare 'flight' domain dictionary using public information. Similar domain specific public knowledge can be used to prepare dictionary for any domain of interest.

Domain Dictionary for Flight Domain: Each flight email contains the information about the departure and arrival airports. In all machine generated emails, these emails not only contain the airport name, but also an IATA code (a unique worldwide code assigned to each airport). IATA codes are 3 or 4 letter alphanumeric codes. We identify that there are around 4000 airports worldwide, from where the commercial flights originate or land (this list is dynamic). Thus, for 'flight' domain, the domain specific dictionary contains the IATA codes and their popular names for all the commercial airports in the world. Further, each flight email must contain the date and time of arrival and departure. After analyzing the flight email data, we included all the common regex patterns used to represent the arrival and departure time and date, as part of dictionary, for each entry in the Domain dictionary, we also store its type. For IATA code, we store the type <CITY>, for a regex corresponding to date, we store its type <DATE> and for a regex corresponding to date-time, we store its type as <DATE-TIME>. Note, we just identify the presence of these keywords in a text node, and not their semantic meaning, i.e., it is not determined yet if a date in a text node is a departure date or arrival date.

Domain Dictionaries bring following advantages: (1) With the aid of domain dictionaries, we narrow down the scope of email. As shown in our experiments, we see up to 98% reduction in the total text that must be considered. Such reduction is highly likely to reduce the requirement of labelled data. (2) Further, we see that the structure of the core region identified based on domain dictionaries and tree-templates is simple as well as repetitive across a diverse corpus of emails. Thus, high precision email annotators, identifying the semantic meaning of the text, can be built if trained on email text corresponding to only tree-templates (building such annotators is outside the scope of this paper).

Domain dictionaries may evolve, i.e., new keywords/type or regex patterns can be included in these dictionaries, if needed, either manually or automatically. Automatic discovery of new keywords for inclusion in the domain dictionary, with the help of *only* the emails processed by our system, is part of our future work.

4 Identifying Core Region

We now present our technique to identify the core region efficiently. While processing the text nodes in the HTML DOM tree of an email, we look for presence of domain specific tokens in it. A node n containing w tokens is assigned a score of w. This score travels all the way from that node up to root of the email DOM tree. The score of each node from node n till root is incremented by w. Therefore, the common parent of two nodes n_1, with score w_1 and n_2 with score w_2 will have a score of $w_1 + w_2$, and so on. Therefore, the root of the tree will have the highest score (w_h). We start traversing back from the root, to find the deepest node from the root, with score of w_h. This node will be the lowest common ancestor in the email tree containing all the of tokens of our interest (no token exists outside this tree). Thus, the sub-tree rooted at the deepest node (farthest from root) with the highest score represents the core region in the email. We call this node, '*C*' and the structure of the tree rooted at *C iTree* (i.e., Information Tree).

4.1 Converting iTree into Template

To convert the *iTree* to a Template, we first replace all those words that match against the domain dictionary by their type. Therefore, 'Alaska' in *iTree* in Fig. 3(a) is replaced by <CITY>, representing the IATA code for that airport, in Fig. 3(b), and so on. Thus, we obtain a structure, as shown in Fig. 3(b).

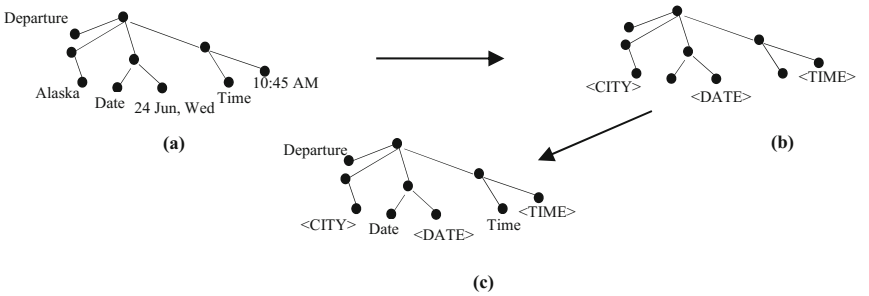


Fig. 3. Template structure in (b) and final template in (c) of iTree in (a)

Before describing our template generation methodology further, we first define similar structure trees.

Def. 4.1.1: Tree Structure Similarity: Two trees T_i and T_j have same tree structure *iff* each non-leaf node in both the trees have exactly same number of children nodes.

Using Def. 4.1.1 and *TreeMatching* algorithm (cf. Sect. 4.2), we collect *N iTrees* instances with matching tree structure. We analyze the distribution of keywords in these instances, to obtain the final template as shown in Fig. 3(c).

If there exists another *iTree*, with similar structure as in Fig. 3(b), but with the keyword '*Departure'* replaced by '*Arrival'*, both these *iTrees* belong to two different templates. Therefore, two instances of *iTree* belong to same template *only if* the tree

structure of the two is same as well as the non-data keywords associated with the *leaf nodes* are also same. Therefore, template discovery process comprises two steps: (1) Clustering instances with similar structure; (2) Among the similar structure instances (Fig. 3(b)), identifying the final templates (as shown in Fig. 3(c)). The algorithm to identify templates is shown below.

Hence, for an input *iTree,* if there exists a template, we update the database with it (function `updatedDB(.)`, line 3, Fig. 4) and take a follow up action, like extracting the actionable information from it (outside the scope of this paper).

```
   Algorithm updateTemplateDB (inputTree iTree, SampleCountThreshold)
1.   int N = SampleCountThreshold;
2.   if ((template= matchTemplate (iTree)) != null)
3.       updatedDB (template, iTree);
4.   else if ((structure=matchStructure(iTree))!= null)
5.       updateStruct (structure, iTree, N);
6.   else
7.       addStruct (iTree);
```

Fig. 4. Update TemplateDB, after processing an email

Now, we describe our approach to extract templates from instances of *iTrees*, in function `updateStruct(.)` (line 5, Fig. 4): First we collect N instances of *iTrees* with same structure. N is set to 40 in our experiments. Let I represents the set of all these *iTree* instances; $|I| = N$. We append a keyword with its DeweyId. For a keyword k, and its DeweyId d, we modify k into $k{:}d$. We compute the frequency profile of these modified keywords (denoted only by k henceforth). We identify the most frequent keywords (frequency cutoff c is a function of N). Let f_k represents the frequency of keyword k. Among the frequent keywords, for a pair of keywords k_1 and k_2 such that $f_{k1} \geq f_{k2}$, if $P(k_2|k_1, I_{k1}) \geq \beta$, k_2 and k_1 become part of the same template, where $I_{k1} \subseteq I$ is a subset of I such that each instance in I_{k1} contains the keyword k_1. β is set high (0.8 or more). If $P(k_2|k_1, I_{k1})$ is less than β but more than α but $P(k_1|k_2, I_{k2}) \geq \beta$, then also, k_1 and k_2 become part of the same template. α is set low (0.2 or less); α, β lie between 0 and 1 ($0 < \alpha < \beta \leq 1$). Hence,

$$P(k_j \mid k_i, I_{ki}) = \frac{\#(k_j \mid k_i, I_{ki})}{\#(k_i \mid I_{ki})}$$

$\#(k_j|k_i, I_{ki})$ is the count of instances in set I_{ki}, where k_j appears given k_i. The templates are updated recursively, till no update happens to any of the templates. Finally, we output the template set S_T; such that $\forall T_i \in S_T; |T_i| \geq c$.

Example: Let the three most frequent keywords in a corpus of N templates be "*Time*", "*Arrival*" and "*Departure*", such that $f_{\text{Time}} > f_{\text{Departure}} > f_{\text{Arrival}}$. Let us assume, "*Departure*" and "*Arrival*" are not part of same template in the ground truth. We check the score P (*"Departure"|"Time", I_{Time}*) first. However, since keyword "*Time*" appears for both "*Departure*" and "*Arrival*", this score is likely to be lower than β (if $\beta = 0.8$, 80% times, "Departure" must occur if "Time" occurs in an instance), assuming

approximately equal distribution of Arrival and Departure emails. However, if the association between the two words is true, the score P ("*Time*"|"*Departure*", $I_{Departure}$) must be high, where $I_{Departure} \subseteq I$ containing keyword "*Departure*" ($|I_{Departure}| \geq c$). Also, as the keywords are appended with DeweyId as well, it ensures that the same keyword appearing at different positions in the *iTree* is treated differently.

If "*Time*" and "*Departure*" becomes part of same template, we next check the score P ("*Arrival*" | "*Time*", "*Departure*", $I_{Departure}$) (i.e., the recursive update of templates). If this score is lower (as will be the case, since "*Departure*" and "*Arrival*" do not occur in same template in the ground truth), the template is not updated further.

Thus, if N instances belong to different templates, we identify them by looking at the keyword distribution in these instances. The identified templates are added in the template DB.

If an email *iTree* structure does not match with any of the existing template structures, we store it as a new tree structure. We identify the underlying templates, as soon as we collect enough emails with similar *iTree* structure from that providers. Therefore, we seamlessly add new templates.

For the identified *templates*, the text nodes of core region can be processed further with the aid of email annotators, i.e., to determine if an IATA code belongs to a departure city or arrival city, etc. These annotators could be supervised or statistical. However, for these annotators, the requirement of the training data reduces significantly if the core region has simpler structure and smaller in size compared to the original email.

Since we ignore rest of the email outside the *iTree,* our method seamlessly accommodates the changes in the email structure outside this core region, thus, further reducing the cost of training data for the supervised email annotators.

4.2 Matching Core Region with Templates

In `matchTemplate()` and `matchStruct()` in line 2 and line 4 in Fig. 4, we match the tree structure of core region *iTree* with a database of existing template trees, to cluster the emails using these template trees. Hence, these functions need a tree matching algorithm. In literature this problem is called approximate subtree matching, [14], and is defined as follows:

Subtree Similarity-Search [12]: Given a tree Q and a database of trees $\Gamma = \{T_1,\ldots, T_n\}$, find the subtrees of trees $T_i \in \Gamma$, most similar to Q.

The size of the template database could be large, containing tens of thousands of templates (as there are thousands of providers for many services, each of which may be generating multiple templates). Thus, it is imperative to match the structure of an *iTree* with an existing template efficiently. Towards that end, we define a problem similar to subtree similarity-search as follows:

Subtree Match Problem: Give a tree T_e and a database of trees $\Gamma = \{T_1,\ldots, T_n\}$, find the trees $T_i \in \Gamma$, which are embedded in T_e.

T_e is the HTML DOM tree of the incoming email. And the database of trees Γ contain all the template trees discovered so far.

We address Subtree Match problem in two steps.

In the first step, we extract the *iTree*(s) of core regions from T_e, as described in Sect. 4 above (note, there can be multiple *iTrees* in an email, as explained in Sect. 5). Thus, the subtree match problem can be reduced to identifying the matching template tree(s) corresponding to discovered *iTree*(s) from a database of templates.

Our next step is to match the *iTree*(s) with existing templates in Γ. Towards that end, for each tree in Γ, we generate its signature such that two trees have same signature *if and only if* their structure is same. Using these signatures, we can efficiently match the trees by comparing their signatures in O (1). Therefore, each unique tree structure must have a unique signature and if two trees have same structures, they must have same signatures.

We exploit the DeweyIds to generate tree signatures as follows: For a discovered *iTree C,* embedded in HTML DOM tree of the email T_e we again generate DeweyIds for the nodes in this tree, considering the root of the *iTree C* as the new root, i.e., the root of the *iTree* is assigned the DeweyID 0. Thus, each node is assigned a new unique DeweyId in the *iTree*. By concatenating the DeweyIds of the leaf nodes in this *iTree*, we generate a string representation of the *iTree*.

We claim, that this string representation of an *iTree* is its unique signature, i.e., if the 'structure' of two *iTrees* T_i and T_j is same, their signatures will be same, and the signature will be different if their structure is different.

Let signature of a *iTree* T_i is denoted by S (T_i).

Lemma (Tree Matching): Two iTrees T_i, T_j have same string representation, i.e., S (T_i) = S (T_j), *if and only if* the structures of these two trees are exactly same.

Proof: Let there be two trees T_1 and T_2 with just one node each. Since the DeweyId of the leafNode of both these trees is same, both the trees generate same string representation, denoted by string '0' (DeweyId of their respective roots/leafNode). Therefore, if the signature of two trees of size one node are same, their tree structure is also same.

Let two trees T_i and T_j, of size n nodes each, have same structure, and thus have the same signature, i.e., each leadNode, in both the trees, has same DeweyId (cf. algorithm ***generateTreeSignature***).

Case1 (Forward Case): We select two leafNodes, in the two trees respectively, with the same DeweyId d. If we add a child node each for both these trees respectively, the child nodes will have the DeweyId of d.0 for both the leafNodes in the two trees. Therefore, based on algorithm ***generateTreeSignature***, the signature of both the trees with (n + 1) nodes remain the same. Hence, signatures of two trees are same, if their structure is same.

Case 2 (Reverse Case): We choose two different leafNodes, with DeweyId d_1 and d_2 in trees T_i and T_j respectively. We add a child node to each of these nodes. The Deweyid of the child node will be d_1.0 and d_2.0 respectively in the two trees, therefore, resulting into two different signatures for two differently structured trees. Therefore, if the structure of two trees is different, their signature will be different. □

With the help of this lemma, we identify the matching template from the template database by a single hash look up. Thus, matching a *iTree* with an existing template is achieved in O (1).

4.3 Issues

However, one issue with this approach is: there could be some sections of the email, which must not be part of core region, but accidentally contain the keywords in the domain dictionary, thus impacting the structure of the core region. In 'flight' domain, some emails may accidentally contain the sequence of characters, forming an IATA code. Thus, having a wrong core region for a fraction of emails may impact the recall of our system (not precision, as there exist no matching template for such emails).

The second issue with this approach is: Our experiments show, even the core regions of the email, carrying same information from the same provider, show significant structural variations. In Table 2 (Sect. 6), we see that 406 emails from Airline 1 resulted in 129 different *iTree* structures for the core region. For Airline 3, it became worse, as 263 emails from this provider resulted in 193 different *iTree* structures (although, these emails were sampled to represent the diversity in the emails from these providers).

It happens because even a minor variation in the HTML of these emails results into a new structure. Our results show, the email structure of even the core region from a single provider is highly dynamic and representing the entire core region through a single *iTree* leads to low recall as even a minor change in email structure may result into a new *iTree* structure, thus failing to match the email with any existing templates. Further, it will also be inefficient as if *iTrees* result into many different structures, we need proportionally more data to identify the correct templates.

5 Improved Method: Sub-templates

We introduce the concept of *sub-templates* to overcome these issues. The basic idea is, instead of representing the entire core region by a single *iTree*, we represent it by multiple *iTrees*, which are called *sub-templates*. Thus, an *iTree* becomes a combination of multiple sub-templates.

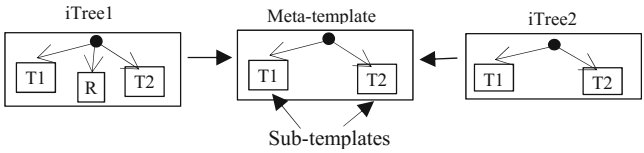


Fig. 5. Extracting meta-templates from iTrees

As shown in Fig. 5, $iTree_1$ and $iTree_2$ are represented as combination of sub-templates T_1 and T_2. Further, $iTree_1$ contains a minor variation in its structure (denoted by node 'R' in Fig. 5), but since it is outside the sub-templates in its tree, region R can be ignored. Hence, both $iTree_1$ and $iTree_2$ can be mapped to a single template. This single template is called the meta-template.

As shown in our experiments (Sect. 6), the concept of meta-templates reduces the number of template structures significantly (as minor variations in the *iTree* are ignored now). We extend this idea further. We consider each meta-template a combination of sub-templates. Thus in the figure above, the meta template is not represented as a single tree, but as $iTree = T_1T_2$, where T_is are the sub-templates. Our results show that meta-templates are in fact permutation of a very small number of sub-templates for a single provider. For example, for Airlines 3, there are 52 meta-templates (representing 193 *iTrees*) but they could be represented using only 4 sub-templates (cf. Table 3).

We get two-fold advantage with sub-templates; (1) the structure of sub-templates is much smaller and simpler; thus, it is easier to build annotators for these sub-templates; (2) we get significantly more instances of each of the sub-templates for each provider, substantially expediting the template discovery process.

Methodology to Identify the Sub-templates: As stated above, each token in a text node of HTML tree of the email, matching against the domain dictionary, is assigned a weight of 1 and this weight travels all the way up to root of the email HTML tree. We identify the lowest common ancestor node of all the tokens and the sub-tree rooted at this node is called *iTree. iTree* contains all the matching tokens. Let us say, this weight of the root of the *iTree* is w_h (i.e., total number of tokens in the email, matching against any domain dictionary token is w_h). We specify a number k, such that $k < w_h$. As a rule of thumb, k must be a small value greater than 1 (usually in the range of 2 to 4). We start traversing the *iTree*, and identify a node n_i in *iTree* such that the weight of n_i is greater or equal to k but there is no node in the sub-tree rooted at n_i, with the weight greater or equal to k. The sub-tree rooted at node n_i represents the sub-template.

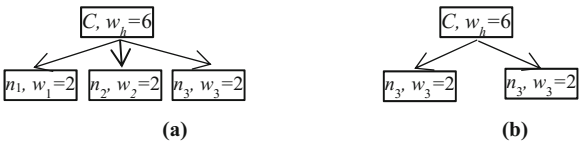


Fig. 6. In (a), sub-template rooted at node C, for k = 3. In (b), sub-templates rooted at n1 and n2, for k = 3

In Fig. 6(a), $w_h = 6$, and $k = 3$. In Fig. 6(a), there is no node in the *iTree* which contains at least k matching keywords in its subtree and root of *iTree* itself is the lowest such node. Hence, *iTree* rooted at node C itself is the sub-template. In Fig. 6(b), two nodes qualify to become the sub-templates and *iTree* is represented as the combination of these two sub-templates. If $k = 2$, the *iTree* in Fig. 6(a) would contain 3 sub-templates.

6 Experiments

We conducted our experiments over two datasets, obtained from the email repository of a large email service provider. First dataset contained flight emails from three different service providers and second dataset contained shipping information emails from Amazon and Walmart. The dataset is anonymized and masked. We construct domain specific dictionaries over these two datasets.

We first present results on how domain specific dictionaries help identify sub-templates, containing information of our interest but much smaller in size compared to original email.

6.1 Discovering Templated Using Domain Dictionaries

In the first set of experiments, we study if the templates extracted with the help of domain dictionaries contain all the information of our interest. We set $\alpha = 0.8$ and $\beta = 0.2$ for all experiments. We also compare the extent of reduction in the template size with respect to original email size.

'Flight' Domain: We prepared the domain dictionary for flight domain as described in Sect. 3.2. In Fig. 7, we show a sub-template extracted from a flight reservation email (passenger names are masked). The original email was 76 KB. The template is less than 2 KB, and contained all the information we sought, thus achieving more than 97% size reduction. Also, we see that the template is much more structured compared to original email which contained a lot of additional information and ads on hotels, car booking, credit cards, etc. As shown in Table 1, we could achieve 91%–98% reduction in the size of email.

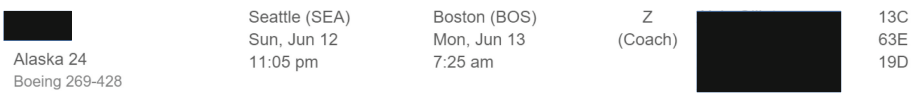


Fig. 7. Core information region of an email, based on sub-template tree

Table 1. Sub-templates size over flight emails

Provider	Avg. mail size (one leg)	Avg. template size	Reduction (%)	Avg. mail size (two legs)	Avg. template size	Reduction (%)
Airline 1	77 KB	2 KB	98%	105 KB	4 KB	96%
Airline 2	32 KB	3 KB	91%	50 KB	4 KB	92%
Airline 3	60 KB	4 KB	93%	64 KB	5 KB	93%

'Shipping' Domain: We analyzed the emails generated by Walmart and Amazon, once the product is shipped after a purchase. We prepared the domain dictionary for 'shipping' domain as follow: Since, our objective is to extract the information about arrival date and the address on which the product would arrive, therefore, the domain dictionary contained several regex patterns for date, zip codes, zip + 4 codes, city names, and state names.

Table 2. Sub-templates size over shipping emails

Provider	Avg. mail size (one product)	Avg. template size	Reduction (%)	Avg. mail size (two products)	Avg. template size	Reduction (%)
Amazon	19 KB	3 KB	85%	30 KB	4 KB	86%
Walmart	136 KB	4 KB	97%	152 KB	5 KB	97%

Table 2 shows that the size of sub-templates, containing relevant information, is smaller by 85% to 97%, compared to original emails.

We show that the domain dictionaries can be built across different domains which help us extract core information regions (represented as template-trees) from the emails.

6.2 Templates and Sub-templates Analysis

Next, we study in detail the templates (*iTrees*) and sub-templates, w.r.t. flight domain. The flight corpus had a diverse set of flight emails, with single leg flights, multiple legs flight, and many other variations in the email. We present our results in Table 3.

As shown in Table 3, there are a large number of unique *iTrees* w.r.t. email corpus size, denoting that the core regions of the email have significant variations in their structure. For instance, there are 193 unique *iTrees* for just 263 emails for Airline 3, i.e., most of the emails resulted into their own unique structure. However, when we convert these *iTrees* to meta-templates (as explained in Sect. 5), we significantly reduce the number of unique structures. Finally, we see these meta-templates are combination of a very small number of sub-templates. For instance, we see that 52 meta-templates comprised just 4 sub-templates for Airline 3. Similar pattern is observed for other airlines, as shown in Table 3. This is a significant insight as it shows that core regions of the emails indeed repeats modulo minor variations and sub-templates capture these core regions. Further, the structure of these sub-templates is much simpler and smaller compared to original email. Finally, with the aid of sub-templates our system was able to handle small changes in the emails structure seamlessly.

Table 3. Extracting different templates from airline data

Provider	#Mails	#iTrees	#meta-templates	#sub-templates
Airline 1	406	129	28	10
Airline 2	258	182	25	5
Airline 3	263	193	52	4

Next, we plot the frequency distribution of different templates types, as shown in Fig. 8, for different airlines. We have truncated the templates after *top* 30, sorted by frequency, as the frequency thereafter is 2 or less. As shown in Fig. 8(a), the highest frequency of an *iTree* was less than 50. Except top three *iTrees*, the frequency for rest was less than 10. In fact, the frequency of 99 *iTrees* (out of 129) was just 'one'. Although, frequency of meta-templates is more, but even in that case, the frequency of last 8 meta-templates (out of 28) was less than or equal to 2 (for Airline 1). However,

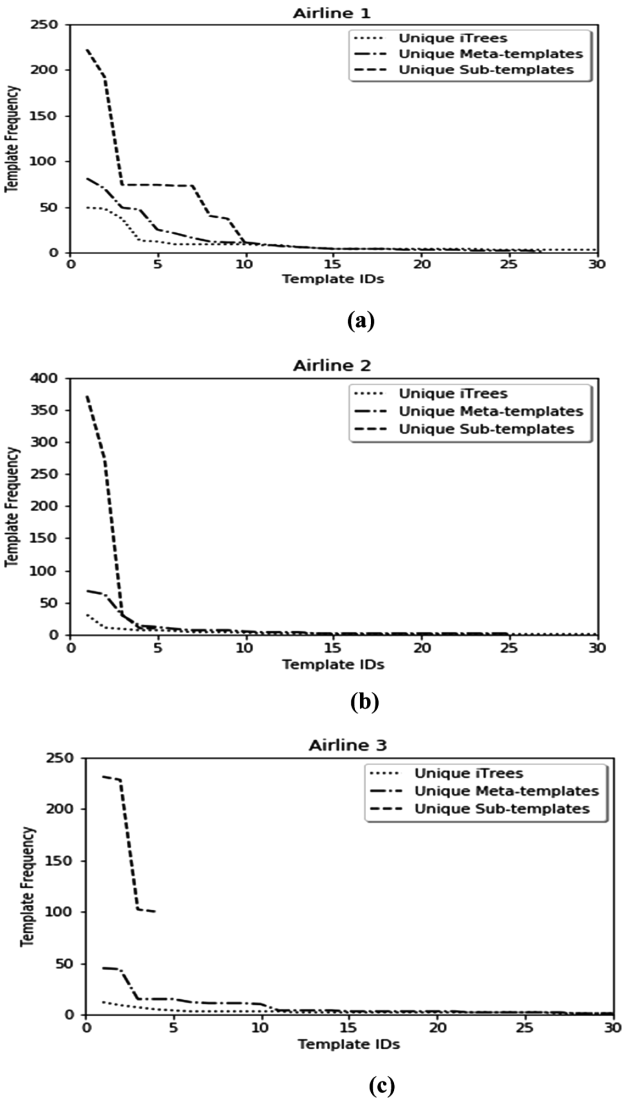


Fig. 8. Sub-templates frequency distribution for flight email data

sub-templates occur with significantly higher frequency. Highest frequency of a sub-template in Fig. 8(a) is more than 200. All sub-templates except one, occur at least 40 times.

In Fig. 8(c), for Airline 3 (corpus of 263 emails), all four sub-templates are present with a frequency of 100 or more. Note, that the highest frequency of a sub-template for Airline 2 is more than the corpus size of 258 mails. The reason is, this template repeats multiple times within the *iTrees* of core regions (due to multiple legs).

We show that our method could extract compact and structured sub-templates. Further, our results on the real data show that the frequency of these sub-templates is high, and the structure simple, enabling building of more accurate annotators.

7 Conclusion

In this paper, we present a novel method to identify core information regions in machine generated emails, such that the core regions contain all the information of our interest. Our results on real data showed that the core region for a diverse set of emails could be represented using only a small number of unique template trees, called sub-templates. Structure of sub-template trees is simple, repetitive in a diverse corpus of emails and their size much smaller compared to original emails. Thus, our system can significantly reduce the need of training data for email annotators. Our future goal is to build an end-to-end email annotator system, that can be extended seamlessly to newer domains. Further, one of our future work item is to automatically expand the domain dictionaries with newer domain specific keywords, *only* with the help of email corpus that our system processes.

References

1. Di Castro, D., et al.: Enforcing k-anonymity in web mail auditing. In: Proceedings of International Conference on Web Search and Data Mining, WSDM 2016, San Francisco, California, USA, pp. 327–336 (2016)
2. Grbovic, M., et al.: How many folders do you really need? Classifying email into a handful of categories. In: Proceedings of International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, pp. 869–878 (2014)
3. Zhang, W., et al.: Annotating needles in the haystack without looking: product information extraction from emails. In: Proceedings of International Conference on Knowledge Discovery and Data Mining, SIGKDD 2015, Sydney, NSW, Australia, pp. 2257–2266 (2015)
4. Liu, B., et al.: Mining data records in web pages. In: Proceedings of International Conference on Knowledge Discovery and Data Mining, SIGKDD 2003, Washington, D.C., pp. 601–606 (2003)
5. Di Castro, D., et al.: You've got mail, and here is what you could do with it! Analyzing and predicting actions on email messages. In: Proceedings of International Conference on Web Search and Data Mining, WSDM 2016, San Francisco, California, USA, pp. 307–316 (2016)

6. Wendt, J.W., et al.: Hierarchical label propagation and discovery for machine generated email. In: Proceedings of International Conference on Web Search and Data Mining, WSDM 2016, San Francisco, California, USA, pp. 317–326 (2016)
7. Zhang, A., Garcia-Pueyo, L., Wendt, J.B., Najork, M., Broder, A.: Email category prediction. In: Proceedings of International Conference on World Wide Web Companion, WWW 2017, Perth, Australia, pp. 495–503 (2017)
8. Maarek, Y.: Is mail the next frontier in search and data mining? In: Proceedings of International Conference on Web Search and Data Mining, WSDM 2016, San Francisco, California, USA, p. 203 (2016)
9. Proskurniay, J., et al.: Template induction over unstructured email corpora. In: Proceedings of International Conference on World Wide Web, WWW 2017, Perth, Australia, pp. 1521–1530 (2017)
10. Avigdor-Elgrabli, N., et al.: Structural clustering of machine generated mail. In: Proceedings of ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, Indiana, USA, pp. 217–226 (2016)
11. Ailon, N., Karnin, Z.S., Liberty, E., Maarek, Y.: Threading machine generated email. In: Proceedings of International Conference on Web search and Data Mining, WSDM 2013, Rome, Italy, pp. 405–414 (2013)
12. Cohen, S., Or, N.: A general algorithm for subtree similarity-search. In: IEEE International Conference on Data Engineering, ICDE 2014, Chicago, Illinois (2014)
13. Tatarinov, I., et al.: Storing and querying ordered XML using a relational database system. In: Proceedings of International Conference on Management of Data, SIGMOD 2002, Madison, Wisconsin, pp. 204–215 (2002)
14. Guha, S., Jagadish, H.V., Koudas, N., Srivastava, D., Yu, T.: Approximate XML joins. In: Proceedings of International Conference on Management of Data, SIGMOD 2002, Madison, Wisconsin, pp. 287–298 (2002)
15. Furche, T., et al.: DIADEM: thousands of Websites to a Single Database. Proc. VLDB Endow. **7**(14), 1845–1856 (2014)
16. Dalvi, N., Kumar, R., Soliman, M.: Automatic wrappers for large scale web extraction. Proc. VLDB Endow. **4**(4), 219–230 (2011)
17. Arso, A., Garcia-Molina, H.: Extracting structured data from web pages. In: Proceedings of International Conference on Management of Data, SIGMOD 2003, pp. 337–348, San Diego, California (2003)

Parameter Free Mixed-Type Density-Based Clustering

Sahar Behzadi[1(✉)], Mahmoud Abdelmottaleb Ibrahim[1], and Claudia Plant[2]

[1] University of Vienna, Vienna, Austria
{sahar.behzadi,a1309720}@univie.ac.at
[2] ds:UniVie, University of Vienna, Vienna, Austria
claudia.plant@univie.ac.at

Abstract. Nowadays many applications generate mixed data objects consisting of numerical and categorical attributes. Simultaneously dealing with mixed objects is more challenging and various approaches convert one type to another one to face this issue. But in many cases this leads to information loss. Therefore integrating categorical and numerical attributes sounds reasonable since it keeps the original format of any attribute. In this paper we focus on clustering and especially density-based clustering as one of the well-known clustering approaches well-performed on arbitrary shape clusters. Density-based clustering algorithms require a distance measure to discover dense regions. Therefore we introduce the distance hierarchy as a distance measure appropriate for both categorical and numerical attributes. However setting the parameters regarding any parametric clustering algorithm could be another issue. Therefore we employ minimum description length principle to automate this process.

Keywords: Density-based clustering · Distance hierarchy
Parameter free clustering · Minimum description length

1 Introduction

Clustering is one of the various data mining tasks which groups most similar data objects together. Many well-known clustering algorithms (e.g. K-means [10] or DBSCAN [11]) measure the euclidean distance as a similarity measure in the sense that the closest object to a specific object is the most similar one. Although this approach sounds reasonable for pure numerical data, considering a mixture of categorical and numerical attributes might challenge its efficiency. However many applications generate a mix of data objects consisting of numerical and categorical attributes.

It is already well-understood that converting one type to another one is not sufficient since it might lead to information loss. Moreover relations between values such as a certain order are artificially introduced. For instance, assuming various regions such as China or United States one can't really define an order or the distances between them.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 19–34, 2018.
https://doi.org/10.1007/978-3-319-98812-2_2

In the meantime integrating categorical and numerical attributes without any conversion seems reasonable since it keeps the original format of any attribute. Considering the fact that almost always there is a natural hierarchy regarding categorical values we introduce distance hierarchy as a distance measure available for both types of attributes. A distance hierarchy extends the concept hierarchy by associating a weight to any link [8]. Also one could assume a distance hierarchy corresponding to any numerical attribute resulting in the euclidean distance.

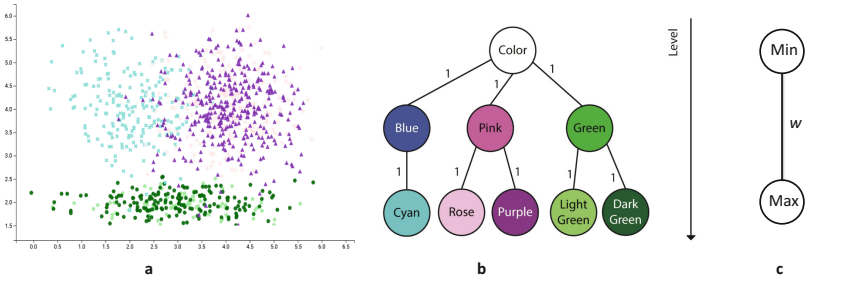


Fig. 1. Synthetic dataset. (a) Three generated clusters with two numerical and one categorical attributes (color). (b) A natural hierarchy between colors. (c) A distance hierarchy corresponding to numerical attributes. (Color figure online)

Figure 1a illustrates a generated dataset comprised of two numerical attributes showing the position of each object and a categorical attribute containing several colors. With respect to the natural hierarchy among various colors Fig. 1b shows the corresponding distance hierarchy to the categorical attribute Color while labels are related to the weights. In this example we assume the same weight for all the links however one could assign different weights due to more information on dataset. To compute the distance between categorical values we utilize the distance hierarchy in the sense that a distance hierarchy provides insights of objects. For instance Rose and Purple are more similar than Rose and Cyan w.r.t. the distance hierarchy. However it is confirmed by the nature of colors since Rose and Purple are derivations of Pink. Preserving the same structure Fig. 1c depicts a distance hierarchy corresponding to the numerical attribute in this example. It has only two nodes and returns the euclidean distance as the distance between two numerical values.

By profiting the distance hierarchy we introduce a general framework appropriate for clustering algorithms which need a distance measure as one of the prerequisites. There are many existing clustering approaches e.g. partition-based, density-based, hierarchical clustering to mention a few. In between density-based algorithms are well-known due to their performance on different (even arbitrary shaped) datasets. DBSCAN is one of the most effective representatives for this approach which captures dense groups of objects as clusters. The basic idea is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster. In this paper we select DBSCAN to compare results of

the proposed framework with state-of-the-art algorithms dealing with mixed-data types.

DBSCAN requires two parameters, a positive real number ϵ and a natural number MinPts showing the radius and the density of a neighborhood respectively. Although DBSCAN is well-known due to its performance, setting appropriate parameters that meet all the aspects of a dataset could be challenging. To face this challenge we propose a parameter free approach by means of *Minimum Description Length* (MDL) principle which links the best clustering with the strongest compression of data.

In this paper we develop a parameter-free mixed-type clustering algorithm modifying DBSCAN which is based on an optimization strategy utilizing MDL. Our contributions consist of:

- **An integrated framework:** We introduce distance hierarchy as a distance measure suitable for both categorical and numerical attributes in the sense that we integrate both types.
- **DBSCAN for mixed-data:** We modify DBSCAN, a well-known density-based clustering algorithm, so that it is applicable for mixed-type data.
- **Parameter-free clustering:** Utilizing MDL principle we introduce a fast noise-robust algorithm without specifying parameters.

In Sect. 3 we introduce the problem specification and introduce a framework suitable for mixed datasets by means of distance hierarchies. In the following we modify DBSCAN in Sect. 4. Section 5 defines a non-parametric version of MDBSCAN following principles of MDL. Finally in Sect. 6 we evaluate our algorithm comparing to others.

2 Related Work

Nowadays to analyze many real applications one need to deal with mixed-type data represented by numerical and categorical attributes. For example, the approaches K-Means-Mixed (KMM) [1], k-Prototypes [9], INCONCO [15], Integrate [3], CFIKP [18], CAVE [7], DH [4] as well as CEBMDC [19].

Most of these approaches use the algorithmic paradigm of k-Means. Often, e.g. in k-Prototypes, not only the number of clusters k, but also the weighting between numerical and categorical attributes should be specified.

The algorithm KMM needs the number of clusters k as input parameter but avoids weighting parameters by an optimization scheme learning the relative importance of the single attributes during runtime. To avoid the difficulty of estimating input parameters, information-theoretic approaches have been proposed. These algorithms (e.g. INCONCO and Integrate) are based on the idea of data compression. The cluster model of these algorithms comprises joint coding schemes supporting numerical and categorical data. The MDL principle allows balancing model complexity and goodness-of-fit. While Integrate has been designed for general integrative clustering, INCONCO also supports detecting mixed-type attribute dependency patterns. The algorithm DH [4] proposes

a hierarchical clustering algorithm using a distance based on the concept hierarchy which facilitates expressing the similarity between categorical values and also unifies distance measuring of numerical and categorical values.

3 Mixed-Data Framework for Clustering

Clustering is the task of grouping different objects of a dataset $\mathcal{DB}$ into k clusters $\mathcal{C} = \{C_1, C_2, ..., C_k\}$. Objects in the same group (cluster) are more similar to each other than to those in other groups (clusters). As mentioned before there are many efficient clustering algorithm that require a distance measure as one the prerequisites. However finding an appropriate distance measure applicable for both categorical and numerical values at the same time is not a trivial task. In this section we introduce a distance measure based on the natural concept hierarchy related to any attribute. A distance hierarchy avoids loss of information by preserving the natural original orders.

Considering a mixed-type data we assume an object O consists of m categorical attributes $\mathcal{A} = \{A_1, A_2, ..., A_m\}$ and d numerical attributes $\mathcal{X} = \{X_1, X_2, ..., X_d\}$. For a categorical attribute A_i, we denote its domain by $Dom(A_i)$ and different categorical values by ${A_i}^j$. According to the natural hierarchy within categorical or numerical values we assume a distance hierarchy corresponding to any attribute. In the following we define the distance hierarchy and introduce a general framework for clustering.

3.1 Distance Hierarchy

Basically a concept hierarchy (tree) consists of concept nodes and links, in which leaf nodes represents more specific concepts while parent nodes are general concepts. Regarding the i-th attribute a distance hierarchy, denoted by $DH_i = (N, \mathcal{E}, W)$, extends the corresponding concept tree by associating a weight to each link. The definition of distance hierarchy in this paper is inspired by [8].

A DH_i has the following properties:

1. DH_i consists of a set of nodes $N = \{n_1, n_2, ..., n_s\}$ and a set of edges $\mathcal{E} = \{e_1, e_2, ..., e_{(s-1)}\}$, where n_j is a parent of n_z if $(n_j, n_z) \in \mathcal{E}$. W denotes the set of weights assigned to any link to facilitate the computation of distance between values.
2. Each node n_j is a concept and represents a sub-category of its parent. Usually data objects are distributed in leaf level. The root node represents associated attribute respectively.
3. The level $l(n_j)$ of a node n_j is the height of the descendant sub-tree. If n_i is a leaf node (e.g. categorical values), then $l(n_j) = 0$. The root node is the attribute A_i which has the highest level, also called the height of the concept hierarchy.

There are many different ways to assign link weights of a distance hierarchy [6,13]. For simplicity in this paper we assign uniformly a constant weight to all links. Other alternatives and a complete investigation on weight assignment approaches is an interesting issue deserving further research in the future.

Let $x = (N_x, d_x)$ denote a point in a distance hierarchy comprising an *anchor* and a positive real value *offset* represented by N_x and d_x respectively. The anchor is a leaf node and the offset represents the distance from the root to x. In the following we explain how to compute the distance between any two categorical or numerical values.

A DH_i regarding to a numerical attribute X_i consists of only two nodes, a root Min and a leaf Max (e.g. Fig. 1c). The associated link weight w equals to the range of X_i, i.e., $w_i = (max_{X_i} - min_{X_i})$. Let $p = (Max, d_p)$ denote a point in a numerical distance hierarchy. Therefore the anchor is always Max and the offset d_p is the distance from the point to the root Min.

3.2 Distance Function and Framework

To clearly define the distance function we need the following definitions:

Definition 1. ***Ancestor.*** *A point p is an ancestor of q if p is one of the nodes existing on the path from q to the root in the corresponding distance hierarchy.*

Definition 2. ***Lowest Common Ancestor (LCA).*** *Considering two nodes p and q in a distance hierarchy the lowest common ancestor or $LCA(p, q)$ is defined as p if p is an ancestor of q otherwise the deepest tree node that is an ancestor of p and q.*

Definition 3. ***Lowest Common Point (LCP).*** *If $p = q$ the lowest common point or $LCP(p, q)$ is defined as p (or q) otherwise $LCA(p, q)$.*

Now we are well-equipped to introduce the distance function. Let $dist(p, q)$ denote the distance between two points p and q w.r.t. the distance hierarchy where p and q could be either categorical or numerical values.

$$dist(p, q) = d_p + d_q - 2d_{LCP(p,q)} \tag{1}$$

where $LCP(p, q)$ is the lowest common point of p and q according to the distance hierarchy and $d_{LCP(p,q)}$ is the distance between the least common point and the root.

Figure 2 depicts an example how to compute the distance between two points $W = (Rose, 2), X = (Purple, 2)$ by means of the distance hierarchy corresponding to the categorical attribute color. W and X belong to the same category and as illustrated in Fig. 2a Pink is the lowest common ancestor of W and X. Therefore $LCP(p, q) = LCA(p, q)$ and $d_{LCP(p,q)} = 1$ w.r.t. Equation 1 and finally $dist(X, W) = |2 + 2 - 2 * 1| = 2$.

However considering $Y = (Cyan, 2)$ and $W = (Rose, 2)$ existing in different categories they naturally should have bigger distance which is approved by our

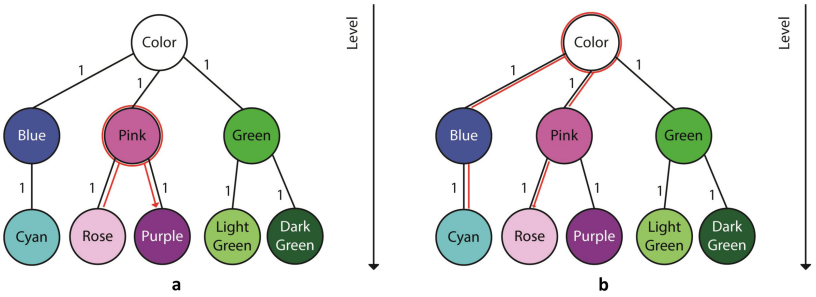


Fig. 2. Computing the distance between colors. (a) Distance between Rose and Purple, $LCA(Rose, Purple) = Pink$. (b) Distance between Cyan and Rose, $LCA(Rose, Cyan) = Color$. (Color figure online)

distance metric. Looking at Fig. 2b one could easily find the least common point of Y and W which is the root node (Color). Therefore the distance between W and Y is $|2 + 2 - 2 * 0| = 4$.

To introduce a general framework for clustering we require mapping objects to distance hierarchies. As mentioned before any attribute of a data object is associated with a distance hierarchy. Let $o = [o_1, o_2, ..., o_n]$ denote an object with n categorical and numerical attributes and let $DH = \{DH_1, DH_2, ..., DH_n\}$ be the set of distance hierarchies associated to any attribute. o_i could be either a categorical value belonging to $Dom(A_i)$ or a numerical value. A categorical attribute A_i associates with DH_i in the way that the set of domain values of A_i corresponds to the leaf nodes of DH_i. As explained before a numerical attribute X_j associates with DH_j which is a degenerated hierarchy (See Sect. 3.1).

Any attribute value o_i is mapped by means of a mapping function h_i to a point in its associated distance hierarchy. For instance let o_i be a categorical value then the mapping $h_i(o_i)$ maps o_i to a leaf node $p = (o_i, d_{o_i})$ in the corresponding distance hierarchy DH_i. For a numerical value o_j the mapping $h_j(o_j)$ maps o_j to $p = (Max, o_j - Min_j)$ in DH_j.

Finally the distance between two mixed-type objects $o_1 = [o_{11}, o_{12}, ..., o_{1n}]$ and $o_2 = [o_{21}, o_{22}, ..., o_{2n}]$ is measured as follows:

$$d(o_1, o_2) = \left(\sum_{i=1,n} w_i (o_{1i} - o_{2i})^L \right)^{1/L} = \left(\sum_{i=1,n} w_i (h_i(o_{1i}) - h_i(o_{2i}))^L \right)^{1/L} \tag{2}$$

where by $L = 2$ the distance is similar to a weighted Euclidean distance.

4 Mixed-Type Density-Based Algorithm

During the previous section we introduced a framework to map a mixed-type object to a point in the associated distance hierarchy and finally we defined a

distance function to compute the distance between objects. As mentioned before the proposed framework is a general framework applicable for any clustering algorithm dealing with mixed-type datasets while it requires a distance metric inside its algorithm. For instance k-means algorithm [10], an efficient clustering algorithm to find Gaussian clusters, assign any object to the closest centroid in terms of distance between the point and any centroid. However applying such algorithms on mixed data types requires either converting attributes which lead to information loss or investigating an appropriate distance metric for both categorical and numerical values. Utilizing the proposed framework enables us to apply an efficient clustering algorithm, designed for pure types of attributes, for a hybrid case.

There are many efficient clustering algorithms dealing with only one type of attributes. In this paper we focus on density-based approaches due to their performance on different datasets (even arbitrary shaped). Particularly we modify DBSCAN [11] and call it **MDBSCAN**. DBSCAN is one of the well-known representatives for this approach which captures dense groups of objects as clusters. The basic idea is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster.

More specifically, DBSCAN needs a positive value ϵ showing the radius of a neighborhood around a point p and the minimum number of points in this ϵ-neighborhood denoted by $MinPts$. Then we start from a random point, find all the points within its ϵ-neighborhood then if the number of those points are bigger than $Minpts$ we build a cluster and keep adding points to it by considering the same procedure for each point in this neighborhood.

To capture the points belonging to the ϵ-neighborhood of an object p we need to compute the distance between all other objects w.r.t. p. This is exactly where our framework plays a role so that DBSCAN would be applicable for mixed-type datasets while Euclidean distance is not a suitable distance any more.

5 Parameter Free Clustering Algorithm

As explained in Sect. 4 DBSCAN requires two parameters, a positive value ϵ showing the radius of a neighborhood and the minimum number of points in this ϵ-neighborhood denoted by $MinPts$. Selection of a higher $Minpts$ leads to more dense clusters. Simultaneously a smaller ϵ forces points to be closer to each other so that they could be considered as a part of a cluster. Although DBSCAN is an efficient clustering algorithm, its efficiency highly depends on parameters while they are hard to specify in advance. Therefore investigating an approach to make this algorithm parameter-free tends to a more effective DBSCAN.

We regard this challenge as a data compression problem applying the principle of Minimum Description Length (MDL). In the following we explain how to make MDBSCAN parameter-free by utilizing (MDL) principle.

5.1 Minimum Description Length (MDL)

MDL is a well-known principle for estimating statistical informations and compressing of data. Regarding clustering as a data compression problem allows us a unifying view, naturally balancing the influence of categorical and numerical attributes in clustering. MDL allows integrative clustering by relating the concepts of likelihood and data compression. To maximize the data compression we assign a shorter description length to regular data objects and longer descriptions to outliers w.r.t. a coding scheme. Following the MDL principle [16], we encode not only the data but also the model itself and minimize the overall description length. The less number of clusters exist in a model the weaker the model fits to the data. On the other side, a model with more clusters tends to be more complex, but has a better fit to the data [3]. The MDL principle finds a natural trade-off between model complexity and goodness-of-fit and thereby avoids over-fitting. In this paper we refer to clustering results as a model associated with data. After clustering to find the compression measure or description length regarding the model we apply MDL as follows:

Definition 4. ***Description Length.*** *Let $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ denote the clustering result consisting of k clusters. The overall description length (DL) of the dataset DB is defined as:*

$$DL(\mathcal{DB}) = \sum_{C_i \in \mathcal{C}} DL(C_i)$$

where $DL(C_i)$ denotes the description length w.r.t. the cluster C_i and is defined as:

$$DL(C_i) = DL_c(\mathcal{X}) + DL_c(\mathcal{A}) + DL(model(C_i)) \tag{3}$$

The first two terms represent coding costs necessary for encoding the numerical and categorical attributes respectively using a specific coding scheme. The last term is the cost of model encoding.

As mentioned we need a coding scheme to find the coding or description length corresponding to any cluster. Huffman coding is one of the well-known coding schemes where the description length of a value o_i is defined by:

$$PDF(o_i).log_2 PDF(o_i)$$

where PDF stands for *Probability Distribution Function*. The output can be interpreted as the numbr of bits necessary to transfer information from a sender to a receiver via a communication chanel. Since the PDF is part of the codebook, any distribution function can be applied [2].

5.2 Coding Numerical Values

Since any distribution function is applicable in the proposed coding scheme and since selection of a specific PDF is not a severe restriction, for simplicity we select Gaussian distribution to illustrate numerical attributes. Thus, for any numerical attribute $X_i \in \mathcal{X}$ we assume $PDF(X_i) = \mathcal{N}(\mu_i, \sigma_i)$. More precisely for a numerical value x the Gaussian probability distribution function is defined as follows:

$$PDF(x) = \frac{1}{\sigma_i\sqrt{2\pi}} exp\left(-\frac{(x-\mu i)^2}{2\sigma i^2}\right).$$

where μ_i and σ_i are mean and variance computed by means of data objects in cluster C_i. Therefore the coding cost corresponding to numerical attribute X_i is:

$$DL(X_i) = -\sum_{x \in X_i} PDF(x).\log_2 PDF(x)$$

Finally based on Huffman coding scheme the first term of Eq. 3 (numerical coding cost) w.r.t. the cluster C_i is provided by:

$$DL_{C_i}(\mathcal{X}) = \sum_{X_i \in \mathcal{X}} DL(X_i)$$

5.3 Coding Categorical Values

The proposed Huffman coding scheme is applicable on categorical attributes as well. However we consider the frequency of any categorical value (leaf nodes in a distance hierarchy) as the associated probability distribution function to the categorical attribute. More precisely considering a categorical attribute $A_i \in \mathcal{A}$ then for any categorical value $A^j{}_i \in Dom(A_i)$ we define $PDF(A^j{}_i)$ in a cluster C_z as $\frac{|A^j{}_i|}{|C_z|}$. Thus, based on the coding scheme the cost of coding categorical attributes in a specific cluster C_z is defined as:

$$DL_{C_i}(\mathcal{A}) = \sum_{A_i \in \mathcal{A}} \sum_{A^j{}_i \in A_i} -PDF(A_i{}^j).\log_2 PDF(A_i{}^j)$$

5.4 Coding the Model

As mentioned considering MDL principles we encode not only the data but also the model itself and minimize the overall description length. So far we have explained how to encode the data consisting of two parts; categorical and numerical part. In this section we elaborate how to encode the model associated with the data so that, back to our example, a receiver has enough information to decode the data. Thus decoding the model one needs to know to which cluster

an object belongs and which parameters specify the cluster model. The first concept is called cluster id denoted by $IDCost$ and the last one is parameter cost denoted by $ParamCost$. The $IDCost$ follows the principle of Huffman coding which implies that we assign shorter bits to the larger clusters. Therefore the cluster id cost w.r.t. the cluster C_z is provided by $\log_2 \frac{|\mathcal{DB}|}{|C_z|}$ where $|C_z|$ shows the number of objects in cluster C_z.

Following the theory of MDL [17] and focusing on a specific cluster C_z the parameter cost to model all the objects in this cluster can be approximated by $\frac{P_z}{2}.\log_2|C_z|$. P_z denotes the number of parameters in cluster C_z required to encode the model. We already assumed a Gaussian distribution to model numerical attributes. Therefore any numerical attribute X_i is described by two parameters: μ_i and σ_i. Regarding categorical attributes we need to encode all the probabilities corresponding to categorical values. Therefore for an attribute A_i with $|A_i|$ categorical values the number of parameters required to be coded is $|A_i| - 1$.

Algorithm 1. Parameter free MDBSCAN

```
Min − DL :=0;
foreach radius ϵ ∈ range R do
    result = MDBSCAN(ϵ,4);
    DL := Compute the description length for results;
    if (DL < Min − DL) then
        Min − DL= DL;
        BestResult = result;
return BestResult;
```

5.5 Algorithm

As explained MDBSCAN requires two parameters to be specified: ϵ and $MinPts$. Referring to original DBSCAN algorithm [11] authors employ $k - dist$ graph to find the best parameter setting. However they claim that for $k > 4$ the $k - dist$ graphs do not differ significantly from the $4 - dist$ graph. Therefore they recommend to set the $MinPts$ as 4. In this paper we stay with the same strategy and fix $MinPts$ as 4 but varying the radius of a ϵ-neighborhood.

Algorithm 1 summarizes our proposed non-parametric MDBSCAN algorithm. Considering $MinPts = 4$ we apply MDBSCAN iteratively for a specific range of ϵ. At the end of each iteration the overall description length DL will be computed following the principle of MDL. Thus for various parameter setting (models) we achieve a comparison score by means of DL. Finally, at the end of iterations we select the parameter setting with the minimum DL i.e. the most compressed model resulting the best clustering.

6 Evaluation

To assess the efficiency and effectiveness of our non-parametric MDBSCAN we compare our proposed algorithm to state-of-the-art mixed-type clustering

algorithms. We selected K-Means-Mixed (KMM) [1], INCONCO [15] and DH [8] so that we can cover all aspects of MDBSCAN. In this section extensive experiments on synthetic and real datasets will demonstrate the advantages of MDBSCAN over other clustering algorithms. All algorithms are implemented in Java and the source code as well as the datasets are available here: https://tinyurl.com/ybqq35xc.

Normalized mutual information (NMI) [12] is an information theoretic evaluation measure for clustering results. In this paper we employ NMI to assess our algorithm in comparison to others. NMI numerically evaluates pairwise mutual information between ground truth and resulted clusters and continues normalizing by means of the entropy of either original or resulted clusters. NMI scales between zero and one representing a random and a perfect clustering, respectively.

6.1 Synthetic Data Experiments

By synthetically generating various datasets we aim to evaluate MDBSCAN covering different aspects e.g. effectiveness, noise-robustness, scalability (Fig. 3).

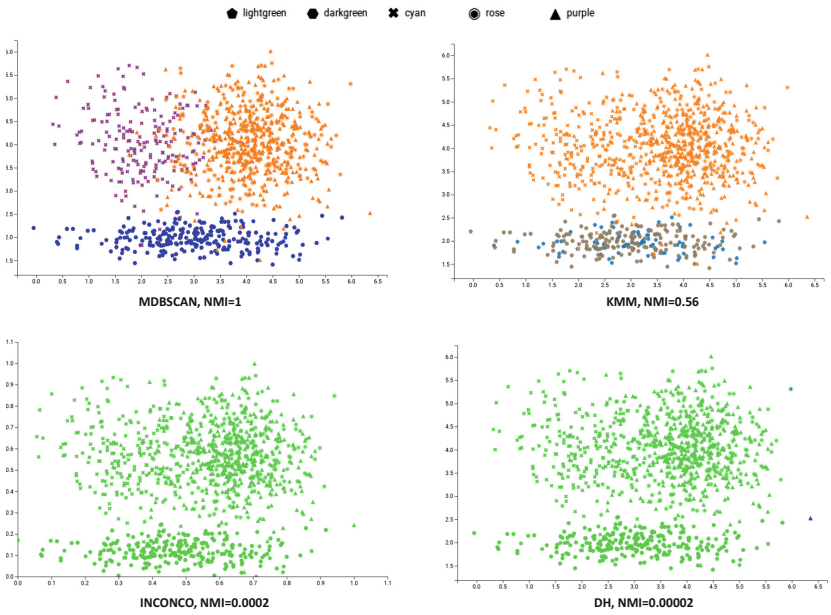


Fig. 3. Clustering results. MDBSCAN, INCONCO, KMM as well as DH.

– **Effectiveness:** In this experiment we consider the same dataset as the running example illustrated in Fig. 1a. Also NMI is applied to assess the effectiveness of a clustering algorithm. As mentioned before a dataset with 3 clusters

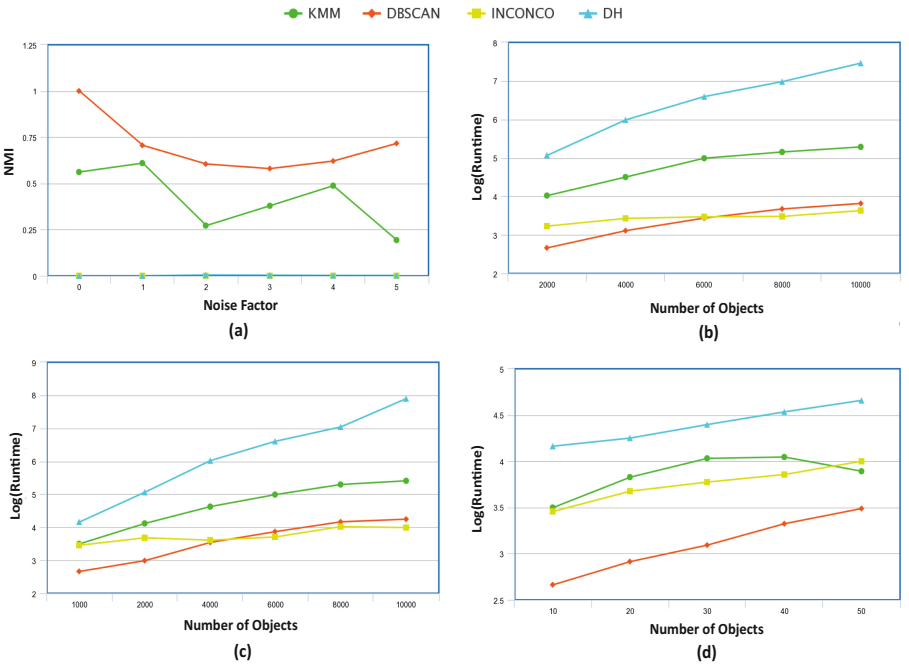


Fig. 4. Comparison in terms of noise-robustness and runtime complexity. (a) Noise-robustness experiment. (b) Runtime experiment by increasing the number of objects while the dimension is 3. (c) Runtime experiment for 10 attributes. (d) Runtime experiment by increasing the dimensionality. (Color figure online)

which consists of 1000 objects with 2 numerical attributes showing the position of any object and a categorical attribute denoting the colors. (Different clusters are illustrated with different shapes in Fig. 1a).
Figure 4 clearly demonstrates that MDBSCAN perfectly outperforms other algorithms in terms of NMI while NMI = 1 shows the best clustering result one could achieve. We illustrate different clusters with different colors to make the comparison more clear. Colors are shown with various shapes. MDBSCAN has been able to assign all Purple and Rose objects to a cluster although some of Purples are positioned in another clusters. (See triangle points in the blue cluster.)

– **Noise-robustness:** To address noise-robustness we introduce a noise factor for both types of attributes. In this experiment we generated a relatively same synthetic dataset as what was generated for the previous experiment. Considering the related distance hierarchy we introduce another category Brown as noise objects distributed in all clusters. We start from 5%. $|C_i|$ noise objects inside any cluster then keep increasing this factor from 1 to 5 times. Moreover for numerical attributes we increase the variance of each numerical attribute by the same factor ranging from 1 to 5 in order to cover

all disturbing aspects. Increasing the variance tends to more mixed clusters. Figure 4a shows the results in terms of NMI running all competitors. As it is clear MDBSCAN outperforms other algorithms regarding any factor of noise while it is non-parametric and finds the exact number of clusters during any experiment.

– **scalability:** We utilize two approaches to address the scalability of our algorithm in comparison to other algorithms. By first approach we increase the number of objects ranging from 2000 to 10000 while the number of attributes is fixed. We considered two different cases in this approach: 1 - the number of attributes is 3 (the data set with almost the same structure as the running example), 2 - the number of attributes is set to 10 (d numerical dimension and 5 categorical). Figure 4a, b indicates that MDBSCAN in both cases is faster than KMM and DH. But in comparison to INCONCO it is faster in the beginning but after almost 5000 objects they have relatively the same run time. However according to the next experiment one could come to the conclusion that MDBSCAN is faster. The other approach is dealing with dimensionality i.e. while the number of objects is fixed the dimensionality is increasing iteratively ranging from 10 to 50. Figure 4d illustrates clearly what we have claimed for MDBSCAN.

6.2 Real Data Experiments

Finally, we evaluate our proposed algorithm MDBSCAN in comparison to other algorithms on real world datasets. As real world problems we used *Teaching Assistant Evaluation* and *Contraceptive Method Choice* from UCI repository [5] and *Airport* dataset from the public project Open Flights [14].

– **Teaching Assistant Evaluation:** The data is provided by [5] and consists of evaluations of teaching performance over three regular semesters and two summer semesters *teaching assistant* (TA) assignments. There are 3 roughly equal-sized categories (low, medium and high) illustrating the scores. The data has 151 objects each of which concerns teaching performance in terms of 4 categorical attributes (Whether the TA is a native English speaker or not, Course instructor, Course, Summer or regular semester) and one numerical (Class size) attribute. In this experiment we consider a flat hierarchy since there is no meaningful hierarchy among categorical values.
  Based on the experimental results MDBSCAN (0.25) outperforms significantly the other algorithms in terms of NMI: INCONCO (0.006), KMM (0.02) and DH (0.02). As mentioned the ground truth is the scores divided to 3 clusters however MDBSCAN found 5 clusters. Although the number of clusters found by MDBSCAN differs from the released labels, it captured characteristics of the dataset better than other algorithms comparing in terms of their mutual information (NMI).
– **Contraceptive Method Choice:** This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey [5]. Samples are married women who were either not pregnant or do not know if they were at

the time of interview. The data consists of 7 categorical and 2 numerical attributes. In this experiment we consider a natural hierarchy for categorical attributes when it makes sense. For instance categorical values corresponding to "Standard-of-living index" consist of 1, 2, 3 and 4. We consider "1" as the lowest standard, "2" and "3" as medium and finally "4" as the highest standard of living. Analogously one could consider the same natural hierarchy for Wife's and Husband's education. For the rest we assume a flat hierarchy.
The target attribute which is used as a ground truth during our experiments is the contraceptive method used by women. This attribute is supposed to group women based on their demographic and socio-economic characteristics. Applying all competitors MDBSCAN (0.35) outperforms other algorithms KMM (0.03), INCONCO (0.04) and DH (0.0001) in terms of NMI.

– **Open Flights Dataset:** The public project Open Flights provides information about airports distributed worldwide. The data consists of 8107 instances each of which has numeric attributes showing the longitude and latitude, the sea height in meters and the time zone. Moreover each object consists of categorical attributes denoting the country, where the airport is located, and the day light saving time. We constructed the concept hierarchy of the country attribute so that each country belongs to a continent. Again three other comparison algorithms (KMM, INCONCO and DH) were applied to this dataset. INCONCO and DH could not find hidden clusters and both of them found only one cluster for this dataset which is not meaningful.
Since there is no ground truth regarding the *Airport* dataset, we first run MDBSCAN and then set the number of clusters required by KMM as the number of clusters found by MDBSCAN in the sense that we could compare them. Figure 5 depicts the discovered clusters after applying MDBSCAN and KMM. MDBSCAN reasonably finds 6 main clusters corresponding to 6 main continents (we consider Russia as European country in distance hierarchy) and two smaller clusters illustrated as roughly noise clusters. However the result of KMM algorithm seems random finding 6 clusters in Asia. For another

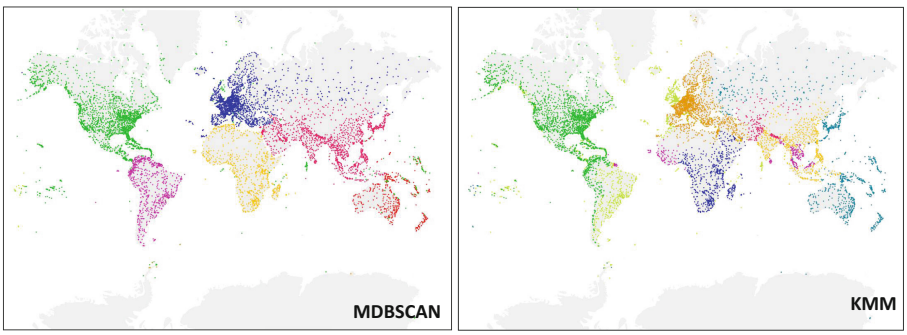


Fig. 5. Clustering results on airport dataset. Comparing MDBSCAN and KMM on Airport dataset.

example KMM groups all the airports located in Australian continent, parts of Eastern Asia and Russia as one cluster while it is hard to interpret this result.

7 Conclusion

To conclude, we introduced an integrative framework to cluster mixed-type datasets consisting of categorical and numerical attributes. In this framework we defined a distance measure, applicable for both types, by means of distance hierarchy. Utilizing this distance measure we avoid converting a data type to another one which tends to information loss and artificially introduced certain orders. Moreover we modified DBSCAN, one of the most efficient and effective density-based clustering algorithm, so that it is able to deal with mixed-type data. Employing MDL principles, we introduced a compression-based approach to score various models and to make MDBSCAN parameter-free. Finally the experiments on synthetic and real datasets indicate the advantages of MDBSCAN in comparison to other state-of-the-art clustering algorithms. However due to Gaussian assumptions considered during the parameter-free procedure may lead to an inaccurate model when the original dataset is non-Gaussian.

References

1. Ahmad, A., Dey, L.: A k-mean clustering algorithm for mixed numeric and categorical data. Data Knowl. Eng. **63**, 503–527 (2007)
2. Böhm, C., Faloutsos, C., Pan, J., Plant, C.: Robust information-theoretic clustering. In: KDD (2006)
3. Böhm, C., Goebl, S., Oswald, A., Plant, C., Plavinski, M., Wackersreuther, B.: Integrative parameter-free clustering of data with mixed type attributes. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS (LNAI), vol. 6118, pp. 38–47. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13657-3_7
4. Hsu, C.C., Chen, C.L., Su, Y.W.: Hierarchical clustering of mixed data based ondistance hierarchy. Inf. Sci. **177**(20), 4474–4492 (2007)
5. Frank, A., Asuncion, A.: UCI machine learning repository (2010). http://archive.ics.uci.edu/ml
6. Das, G., Mannila, H., Ronkainen, P.: Similarity of attributes by external probes. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 23–29 (1998)
7. Hsu, C.C., Chen, Y.C.: Mining of mixed data with application to catalog marketing. Expert Syst. Appl. **32**(1), 12–23 (2007)
8. Hsu, C.C., Chen, C.L., Su, Y.W.: Hierarchical clustering of mixed data based on distance hierarchy. Inf. Sci. **177**(20), 4474–4492 (2007)
9. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. **2**, 283–304 (1998)
10. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1: Statistics, pp. 281–297. University of California Press, Berkeley (1967). https://projecteuclid.org/euclid.bsmsp/1200512992

11. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of KDD 1996. AAAI Press (1996)
12. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: ICML (2005)
13. Palmer, C.R., Faloutsos, C.: Electricity based external similarity of categorical attributes. In: Whang, K.-Y., Jeon, J., Shim, K., Srivastava, J. (eds.) PAKDD 2003. LNCS (LNAI), vol. 2637, pp. 486–500. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36175-8_49
14. Patokallio, J.: Open Flights (2016). http://openflights.org/data.html
15. Plant, C., Böhm, C.: Inconco: interpretable clustering of numerical and categorical objects. In: KDD, pp. 1127–1135 (2011)
16. Rissanen, J.: A universal prior for integers and estimation by minimum description length. Ann. Stat. **11**(2), 416–31 (1983)
17. Rissanen, J.: An introduction to the MDL principle. Technical report, Helsinkin Institute for Information Technology (2005)
18. Yin, J., Tan, Z.: Clustering mixed type attributes in large dataset. In: Pan, Y., Chen, D., Guo, M., Cao, J., Dongarra, J. (eds.) ISPA 2005. LNCS, vol. 3758, pp. 655–661. Springer, Heidelberg (2005). https://doi.org/10.1007/11576235_66
19. He, Z., Xu, X., Deng, S.: Clustering mixed numeric and categorical data: a cluster ensemble approach. CoRR, abs/cs/0509011 (2005)

CROP: An Efficient Cross-Platform Event Popularity Prediction Model for Online Media

Mingding Liao[1], Xiaofeng Gao[1(✉)], Xuezheng Peng[2], and Guihai Chen[1]

[1] Shanghai Key Lab of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
liao.mingding@gmail.com, {gao-xf,gchen}@cs.sjtu.edu.cn
[2] Baidu, Inc., Beijing, China
pengxuezheng@baidu.com

Abstract. The popularity analysis of social media is crucial for monitoring the spread of information, which is beneficial to public concerns track and decision-making for online platforms. Numerous studies concentrate on the trend analysis on single platform, but they neglect the data correlation between different platforms. In this paper, we propose CROP, a cross-platform event popularity prediction model to forecast the popularity of events on one platform based on the information of the auxiliary platform. We first define the cross-platform event popularity prediction problem. Then we clean the data and explore the slot matching of event time series in diverse platforms. Moreover, we first define the aggregated popularity for the feature construction of event popularity prediction model. Finally, extensive experiments based on events data show that CROP achieves great improvement for predicting accuracy over other baseline approaches.

Keywords: Cross-platform · Event popularity prediction
Support vector regression

1 Introduction

In today's world, the Internet has become the main information dissemination media with large user base. Therefore, the analysis of information dissemination in the online media have become an important research topic [7]. The online media are the complex system consisting of diverse platforms including social networks, search engines, online group, online forum, etc. A large number of studies focus on the event popularity analysis on a single platform [11–14,16], but they neglect the data correlation between different online platforms. However, event information is generally spread through the multiple platforms. For example, an event would be introduced on the website, be searched on the search engine, be relaid upon the social network and be talked in the forum. In fact,

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 35–49, 2018.
https://doi.org/10.1007/978-3-319-98812-2_3

particular properties of different platforms, such as the users communities and the speed of propagation, demonstrate the potential for cross-platform research for event popularity prediction, which plays a crucial a role in understanding the process of event propagation [19].

Therefore, we design CROP, an efficient **CRO**ss-**P**latform event popularity prediction model for online media. To measure the event popularity, CROP employs and improves the Term Frequency-Inverse Document Frequency (TF-IDF) based method which is developed in [19]. Then CROP studies the correlation of time series and employs Dynamic Time Warping method and improves it with the penalty and compound distance to generate sequence alignment matches. Next, we give the novel definition of aggregated popularity for the purpose of the application of data correlation and CROP extracts features on the basis of aggregated popularity. Last, Support Vector Regression is employed for event popularity prediction.

The contribution of this paper are summarized as follows: (1) We design an efficient scheme called CROP for cross-platform event popularity analysis and prediction with only post and query information; (2) We first study on and make use of the correlation of the cross-platform data for event popularity prediction; (3) We first improve Dynamic Time Warping method for time series matching and then define the aggregated popularity for feature construction on the basis of time series correlation; (4) We conduct extensive experiments on several datasets, which demonstrate that CROP performs best among the baselines.

The rest of this paper is organized as follows. In the Sect. 2, we introduce the related work of CROP. The problem statement is given in Sect. 3. In the Sect. 4, we introduce the overview and main components of CROP. The experiments are showed in Sect. 5. Finally, the paper is concluded in Sect. 6.

2 Related Work

2.1 Event Popularity Analysis on Individual Platform

A event could be defined as a single word or a coherent set of semantically related terms which summarizes the majority of related documents or other semantics items [12,17,24]. Event popularity in online media is usually defined as the number of posts, reposts queries or comments. There are numerous studies regarding event popularity prediction, and machine learning method is the major approach.

Leskovec et al. designed a meme-tracking approach and studied the coherent representation of the popularity in the new cycle. However, this paper emphasizes the periodicity and did not propose an accurate numerical method [10]. Armed with this research, Yang et al. [23] proposed the K-Spectral Centroid (K-SC) clustering algorithm with the wavelet-based incremental version to solve the problem. Wang et al. [22] also improved the K-Spectral Centroid method by selecting orthogonal polynomial function and using wavelet transformation to decrease the dimensionality of data. Bandari et al. [1] constructed a multi-dimensional feature space and employed regression and classification for popularity prediction. Then Wang et al. [21] aimed at predicting the time of appearance

of the burst and then reduced it into a classification problem. Miao et al. [12] predicted the event popularity based on template vectors of popularity and speeded up the time series prediction method by setting representative users.

2.2 Cross-Platform Popularity Analysis

Few studies noticed the potential of cross-platform popularity analysis and attempted make use of the correlation of cross-platform data. Giummole et al. first studied the trending events in Twitter and Google and found that most Twitter trends would cause the later occurrence of similar Google trends [5]. On the basis of it, the Bipartite Graph of the trend was introduced to handle the similar keywords [6]. Then Tang et al. confirmed that the correlation also existed in Baidu and Sina Weibo [19]. Tolomei et al. noticed that Wikipedia had the Entity Linking technology, which could be linked by Twitter and provided the feasibility to predict the popularity of Wikipedia article based on Twitter data [20]. Keneshloo et al. pointed out the fact that the posts in the social network which mentioned an article could enhance the popularity of the article [8]. Chen et al. paid attention to two commonly used information acquisition platforms: Google and Stack Overflow. This study stated that the correlation between the Google Trends and Stack Overflow Data Dump and provided the dataset for others' future research [2].

3 Definition and Problem Statement

For an event, the raw data should be the semantic relevant items, for example, microblogs in social networks and queries in search engines. The time span of the corresponding event is divided into n periods and $T = [t_1, t_2, \ldots, t_n]$. For different platforms, $\mathbf{R^p}$ is defined as the collection of all information about the event in the platform p. $\mathbf{R_i^p}$ is the collection of all information on the platform p at the time slot t_i and $r_{i,j}^p$ is the j-th records in the R_i^p. In details, we organize the records as a sequence of words: $r_{i,j}^p = \langle d_{i,j,1}^p, d_{i,j,2}^p, \ldots, d_{i,j,s_{i,j}^p}^p \rangle$, where $s_{i,j}^p$ is the length of the sequence.

The event popularity on the platform p is labeled as pop^p, and pop_i^p is the popularity in the platform p at the time slot t_i. CROP provides two different definition of event popularity in terms of text length. On the one hand, the event popularity of short-text platform is defined as Eq. (1). For example, most queries in the search engine are extremely short with only one or two words and the purpose of queries is for acquaintance of events, so all of the queries should be treated equally.

$$pop_i^{p_1} = \frac{|\mathbf{R_i^{P1}}|}{\max_{i=j}^{n} |\mathbf{R_j^{P1}}|} \tag{1}$$

One the other hand, we employ a cross-platform event dissemination trend analysis approach [19] based on TF-IDF method in the long-text platform and make the necessary modification to improve the generality of the model, for example, Twitter. The popularity of an event is measured with the help of the hot

words $HW = \{hw_1, hw_2, \ldots, hw_m\}$, where m is the number of hot words. Thus, the definition of the event popularity in the long-text platform is described as Eqs. (2) (3) (4), where nhw_j^i is the number of occurrences of hot word hw_i at the time slot t_j. In detail, $freq_i^{hw_j}$ is the term frequency of the hot word hw_j at the time slot t_j and $rec_i^{hw_j}$ is corresponding inverse record frequency. Different from the standard TF-IDF algorithm, we only consider the data before the time slot t_j as the total records instead of all data when computing the inverse record frequency to improve the applicability because the total records could be difficult to achieve.

$$pop_i^{p_2} = \sum_{j=1}^{m} rec_i^{hw_j} \times freq_i^{hw_j} \tag{2}$$

$$freq_i^{hw_j} = \frac{nhw_j^i}{\sum_k nhw_k^i} \tag{3}$$

$$rec_i^{hw_j} = \log \frac{|\bigcup_{k \le i} \mathbf{R_i^{p_2}}|}{|\{x | hw_i \in x, \in \bigcup_{k \le i} \mathbf{R_i^{p_2}}\}| + 1} \tag{4}$$

Both two definitions of popularity are independent for data, which ensures the applicability of CROP. Based on the previous definitions, the problem of the cross-platform event popularity prediction could be defined as Definition 1.

Definition 1. *Given the relevant raw documents R^{p_1} and R^{p_2} about a event and the division of the time span T. Provide the most proper values of popularity of the event at the next time slot on the target platform p_1 with the help of data on the assistant platform p_2.*

4 Cross-Platform Popularity Prediction Model

4.1 Overview

Figure 1 illustrates the structure of CROP. CROP consists of five steps: data preprocessing, hot word extraction, popularity analysis, time series matching and popularity prediction.

The first part is data preprocessing. Then the hot words are selected from the preprocessing raw data in the word extraction part. After the calculation of hot word dynamic frequency and the recursive document frequency, the popularity of the event is measured on the two platform at each time slot and is denosing by Discrete Wavelet Transform. In the next step, the two time series are matched by the Dynamic Time Warping method and the aggregated popularity is defined on the basis of the matching of popularity time series. Finally, the novel features constructed from aggregated popularity are used in the Support Vector Regression methods for event popularity prediction.

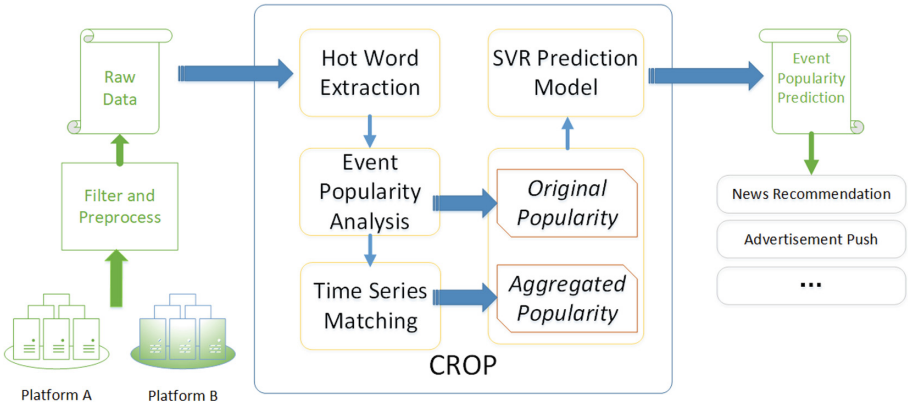


Fig. 1. Overview of CROP

4.2 Data Preprocessing

In the process of data acquirements, we utilize three methods to filter data. First, we provide related words for the corresponding event and only the records containing the given words or the similar words would be selected. Second, considering the enormous size of relevant data in the online media, the datasets used are sampling from the whole data. Last, the meaningless part of the raw data, such as URL, would be filtered.

4.3 Hot Word Extraction and Popularity Analysis

Before designing the model for popularity prediction, we have to provide a popularity measuring method. To make the process of popularity analysis convenient, the word extraction method is employed to extract the words and find the hot words. All of the documents are split into words based on a open-source project called jieba[1]. In the process of word extraction, we filter the stop words, which are the function words without actual meaning.

The hot words are selected based on the result of words extraction by the library in jieba, and then calculate the popularity by the definition mentioned in Sect. 3.

Because of the randomness of user behavior on the Internet, the event popularity series may contain noise. We novelly employ the Discrete Wavelet Transform method to denoise the data of the popularity. We utilize the Python library PyWavelets[2] to implement the process. We employ the Haar wavelet to implement the discrete wavelet transform [18].

[1] https://github.com/huaban/jieba-analysis.
[2] https://pypi.python.org/pypi/PyWavelets/.

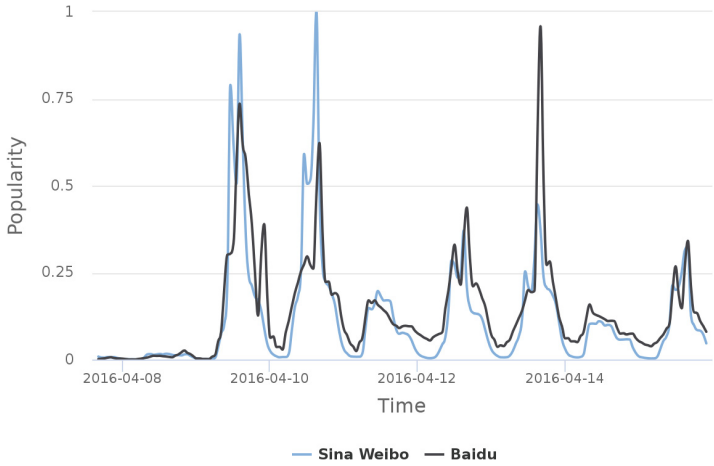


Fig. 2. The example of event popularity

4.4 Time Series Matching

Figure 2 is the popularity result of the event "Lee Sedol v.s. AlphaGo" in Sina Weibo and Baidu, which is introduced in Sect. 5.1, from the 16:00, March 7th, 2016 to the 00:00, March 16th, 2016. It demonstrates that the time series of two platforms are not matched slot by slot. To solve the problem, we novelly employ Dynamic Time Warping (DTW) method which is a popular dynamic programming method in the field of speech pattern recognition [3,15]. Dynamic Time Warping problem could be defined as the following: given two time series $P = \{p_1, p_2, \ldots, p_n\}$, $Q = \{q_1, q_2, \ldots q_m\}$,and find a matching from P to Q with the lowest cost. To simply the problem, it is assumed that each items p_i in P is matched with a continuous subsequence (called $M(p_i)$) of Q.

In term of the actual condition of the cross-platform popularity prediction, an obvious fact should be noticed that the time alignment of the two popularity time series should not be large. Therefore, we propose two ideas to improve the basic form of algorithm of the Dynamic Time Warping: (1) We define the penalty function to improve the similarity calculation based on the time difference; (2) We compress the state space of dynamic programming, which is the main process of Dynamic Time Warping. In the following part of this section, we describe the details of improvement.

Cost Function with Penalty. The cost function with penalty, called as $C(p_i, q_j)$, is the cost to match p_i and q_i. The basic idea is measuring the matching cost by Euclidean distance of p_i and q_j [15]. [9] showed the deviation of the time series could provide simple and robust measure result. Thus, we ulitize the geometric mean to integrate the Euclidean distance of the popularity value and the deviation of that, which is showed in Eq. (5).

$$C_{base}(p_i, q_j) = \sqrt{|p_i - q_i| \times |(p_i - p_{i-1}) - (q_i - q_{i-1})|} \tag{5}$$

Morever, we design the penalty coefficient for the constraint that the two time slots away from each other would not be matched. Logistic function is employed to define the penalty as the Eq. (6), where β is a parameter to control the effort of C_{pena} and b is an expectation of time slot bias.

Based on the Eqs. (5) and (6), the final formula of the cost function with penalty is Eq. (7).

$$C_{pena}(p_i, q_j) = \frac{1}{1 + e^{-\beta \times (|i-j|-b)}} \tag{6}$$

$$C(p_i, q_j) = C_{base}(p_i, q_j) \times C_{pena}(p_i, q_j) \tag{7}$$

Limited State Space. The naive DTW method suffers from high executing time. Limitation of the state space is proposed to solve the efficiency problem, which is inspired from the fact that we need not consider the matching with two time slots away from each other.

$LTC(P_{1..i}, Q, k)$ is defined as the minimal of the total matching cost of $P_{1..i}$ and $Q_{1..(i-k)}$ and the state space could be limited by the value of k. In detail, P is the popularity time series on target platfrom and Q is the popularity time series on assistant platform.

The recursive formula to calculate LTC is showed in Eq. (8).

$$LTC(P_{1..i}, Q, k) = \max \begin{Bmatrix} LTC(P_{1..i-1}, Q, k-1) + C(p_i, q_j) \\ LTC(P_{1..i}, Q, k+1) + C(p_i, q_j) \\ LTC(P_{1..i-1}, Q, k) + 2 \times C(p_i, q_j) \end{Bmatrix} \tag{8}$$

In the first case of Eq. (8), we give the limitation that $i > 0, k > 0$, and $k < len$ in the second case, and $i > 0, i - k > 1$ in the third case. Also, we let $k > 0$ in all cases, because we can only use the data of the assistant platform which is monitored before the predicting time.

To facilitate the following process of CROP, $ORI(P_{1..i}, Q, k)$ is set to record how the $LTC(P_{1..i}, Q, k)$ is calculated. It could assist us to find the previous state conveniently and know the best matching between $P_{1..i}$ and $Q_{1..(i-k)}$. The $ORI(P_{1..i}, Q, k)$ would be set from 1 to 3, which corresponds to the three cases of the origin of $LTC(P_{1..i}, Q, k)$.

4.5 Prediction Model Establishment

CROP reduces the event popularity prediction problem into the regression problem and construct the feature space based on the known event popularity on the target and assistant platforms. However, the different event propagation trends of two platforms inspire the special features. Considering the result of time series matching, the several time slots in the event popularity time serial on the assistant platform could be matched with the same time slot in the event popularity time serial on the target platform. The complicate relationship makes it arduous to employ data correlation for machine learning model. Therefore, CROP

defines novelly aggregated popularity for the feature construction. The aggregated popularity $agpop_j^{p_2}$ on the assistant platform p_2 is defined as the average of the event popularity on p_2 at the time slot which is matched with the time slot t_j on target platform p_1. Algorithm 1 shows the calculation of aggregated popularity.

Algorithm 1. Aggregated Popularity Calculation

Input: popularity time series $P[1..n], Q[1..m]$, the parameters vis
Output: $agpop$ which is the aggregated popularity of Q

Invoke the Dynamic Time Warping method with the input of P, Q and vis and get LTC and ORI;
for $1 = 1; i \leq n; i \leftarrow i + 1$ **do**
  $index \leftarrow i - \arg\min_j(LTC(i, j))$;
  $now \leftarrow i$;
  $direc \leftarrow ORI(\arg\min_j(LTC(i, j)))$;
  for $j \leftarrow 0; j < vis; j \leftarrow j + 1$ **do**
    $tot \leftarrow 0$;
    $sum \leftarrow 0$;
    while $direc = 2$ **do**
      $tot \leftarrow tot + 1$;
      $sum \leftarrow sum + q_{index}$;
      switch *the value of direc* **do**
        case *1*
          do not change $index$;
          $now \leftarrow now - 1$;
          $direc \leftarrow ORI(\arg\min_j(LTC(now, now - direc)))$;
        case *2*
          $index \leftarrow index - 1$;
          do not change now;
          $direc \leftarrow ORI(\arg\min_j(LTC(now, now - direc)))$;
        case *3*
          $index \leftarrow index - 1$;
          $now \leftarrow now - 1$;
          $direc \leftarrow ORI(\arg\min_j(LTC(now, now - direc)))$;
    $pop_{i-j}^{i} \leftarrow sum/tot$;
return pop;

Therefore, when considering the time slot t_i, $pop_i^{p_1}$ is treated as the label where p_1 is the target platform, and $\{pop_{i-vis}^{p_1}, \ldots, pop_{i-1}^{p_1}\}$ can be used as a part of feature, where vis is set to represent the length of time slot used for the feature establishment in the regression model. Moveover, the aggregated popularity $\{pop_{i-vis}^{p_2}, \ldots, pop_i^{p_2}\}$ on the assistant platform also be utilized as features. In all, the feature vector of time slot t_i is $\langle pop_{i-vis}^{p_2}, \ldots, pop_i^{p_2}, pop_{i-vis}^{p_1}, \ldots, pop_{i-1}^{p_1}\rangle$.

After the label and feature construction, the Support Vector Regression model is employed for event popularity prediction, which is one of the most efficient model for regression.

The time complexity of Algorithm 1 is $O(n \times (vis + len))$, where vis is the size of feature space and len is the length of limited state space of dynamic programming. Line 2 and Line 6 enumerates i and j, so the low bound of the complexity is $O(n \times vis)$. The running time from Line 9 to Line 25 is not trivial to derive, but every time the loop executes, either now or $index$ would decrease 1. In fact, now is equal to j and $index$ is the time slot matched with now, so $now \leq index - len$. Thus, for each i, Line 9 to Line 25 is only executed at most $2 \times vis + len$ times. Therefore, the time complexity is $O(n \times (vis + len))$.

5 Experiments

In this section, we present the experimental results of CROP and analyze the effectiveness. All of the experiments are implemented by Python 2.7 and running on a PC with the Intel(R) Core(TM) i5-6500 CPU @ 3.20 GHz, 8 GB RAM on the Ubuntu 16.04 operating system. The experiments are based on the datasets of three hot events in the Baidu and Sina Weibo, which are most popular search engine and social network in China.

5.1 Experiment Setup

Dataset Description. In this paper, we concentrate on the search engine (Baidu) and the social network (Sina Weibo) as the object of research, because they are the most important parts of social media and they have the unique properties. The social network is the platform of initiative information sharing, but the search engine only provides the information based on the users' queries, which lead to the hysteresis of the popularity time series of hot events in the search engine comparing with that in the social network [19]. Thus, the Search Engine is the object platform and the Sina Weibo is the target platform.

The dataset used in our experiment is the data of three hot events in Baidu, which are provided by research institute of Baidu Online Network Technology Company through internal interface and the corresponding data in Weibo through crawler. The datasets of Baidu consist of massive inquire records with inquire words and cryptographic user identity with data masking. The datasets of Sina Weibo consist of massive post records and cryptographic post information. Table 1 shows the details of the datasets. All of the dataset are the hot events from 2015 to 2016 in China. The "Lee Sedol v.s. AlphaGo" event is about the go competition between one of the best human go player and the artificial intelligence AlphaGo. "Brexit" is the event that majority of British people wanted to vote to leave the European Union in the wake of the migrant crisis. The event "The Capsizing of a Big Ship" was started by the accident that a ship called "Easten Star" capsized on Yangtze River with hundreds missing, and became the hot event because the plenties of later news and talks.

Table 1. Overview of the experimental dataset

Topic name	Date	Data of Sina Weibo	Data of Baidu
Lee Sedol v.s. AlphaGo	02/20–03/29 (2016)	654.89 MB	406.83 MB
The Capsizing of a Big Ship	06/01–06/29 (2015)	320.59 MB	401.48 MB
Brexit (Britain + Exit)	06/21–06/30 (2016)	715.51 MB	392.32 MB

For each dataset, we use 1 h as a unit of the time slot and divide them into discrete time series, because 1 h is a common time unit for social content process. For example, Sina Weibo maintains the hot post list by 1 h. To simplify the examination of our model, we arbitrary select a subsequence of the time series in 300 h. The data in the first 200 h is treated as the training set, and the other data is used as testing data.

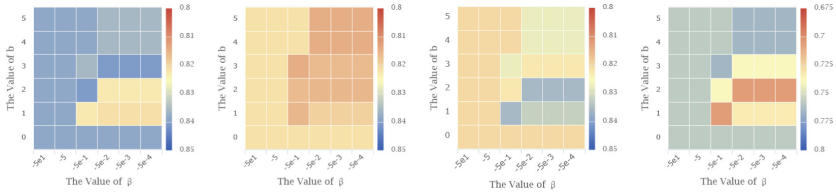
Metric Measures. In our experiments, the adjusted Mean Absolute Error (called $aMAE$) and the R-square (called $R\text{-}square$) are utilized to measure the predicting accuracy of the models. R-square is a common metric in regression model. Adjusted Mean Absolute Error is the metric improved from the Mean Absolute Error. The formula of these two metrics are Eq. (9), where y' is the label value of the item in the testing set and $\hat{y}'$ are the predicting result of y' and $\bar{y}'$ is the average of all y'. The predicting accuracy is higher if the adjust Mean Absoluted Error is smaller and the R-square is closer to 1.

$$aMAE = \frac{\sum_{y'} |y' - \hat{y}'|}{ave} \quad R\text{-}square = \frac{\sum_{y'}(y' - \bar{y}')^2}{\sum_{y'}(\hat{y}' - \bar{y}')^2} \tag{9}$$

Baseline. We set two baselines for experiments. The first baseline is SVR, which is the classical Support Vector Regression only based on the previous vis time slot popularity in the search engine. The second baseline is cro-SVR [6], which uses popularity at the previous vis time slot in the social network.

The difference between cro-SVR and CROP is that cro-SVR does not use the time series matching to eliminate the effect of time series alignment.

Parameters Setup. In this part, we set the value of the primary parameters in these experiments. We complement the preliminary experiments on "Lee Sedol v.s. AlphaGo" datasets. Figure 3 displays the predicting accuracy measured with R-square when selecting different β and b. In the corresponding preliminary experiments, vis is set as 2, 3, 4, 5. The experimental result shows that there is no option of β and b that performs best in all situations. Moreover, comparing Fig. 3(d) with other three figures, we could find the accuracy change in the Fig. 3(a), (b) and (c) is much smaller than that in Fig. 3(c). Therefore, it is vis instead of β and b that would influence the predicting accuracy.



(a) $vis = 2$ (b) $vis = 3$ (c) $vis = 4$ (d) $vis = 5$

Fig. 3. R-square w.r.t. β and b

Thus, it the following experiments, we set $\beta = -0.5$ and $b = 4$, which performs well in most situation, and we would concentrate on the influence of vis to the predicting accuracy.

5.2 Experiment Analysis

Analysis on Datasets. In this part, we would analyze the datasets about three hot event by the V/S Test and the Pearson Correlation Coefficient.

V/S test [4] is a typical method to distinguish whether the time series have long-term memory or short-term memory. The Hurst parameter H is the result of V/S Test. If $H > 0.5$, the sequence has long term memory. If $H \leq 0.5$, the sequence has short term memory.

Pearson Correlation Coefficient is used to reflect the degree of linear correlation between two variables. The greater the absolute value of Pearson Correlation Coefficient(PCC), the stronger the correlation.

Table 2 shows the V/S Test result of the datasets about event hot events the Pearson Correlation Coefficient between the popularity time series on Sina Weibo and Baidu.

Table 2. Statistical Analysis of the Experimental Datasets

Topic name	Hurst of Baidu	Hurst of Sina Weibo	PCC value
Lee Sedol v.s. AlphaGo	0.389	0.379	0.839
The Capsizing of a Big Ship	0.375	0.322	0.762
Brexit (Britain + Exit)	0.422	0.388	0.859

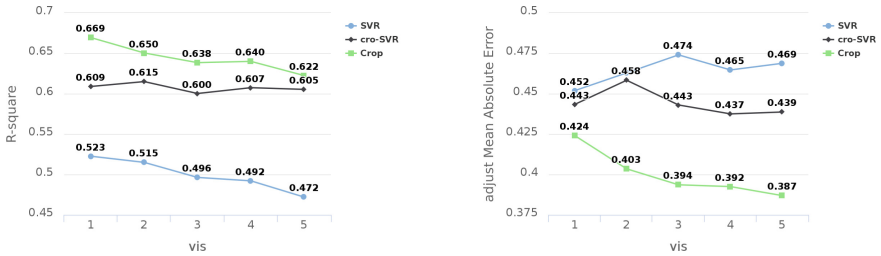
Based on the result showed in the Table 2, we gain an intuitive understanding of about the popularity time series extracted from the datasets. The event popularity time series of hot eventS on Baidu and Sina Weibo have short-term memory, which means that it is reasonable to consider short term data only for event popularity prediction. Moreover, popularity time series of the same event have high correlation, which shows that the idea to predict the popularity of hot events on the search engine based on the data on the social network is feasible.

Experiment Results. In the part, we would provide the experiments on CROP and the other two baseline approaches SVR and cro-SVR in the datasets. The following figures show the experimental results in the event datasets of the hot event. In all, the performance of CROP is better than the other two baseline methods. In the following part, we will explain the experimental results in detail and analyze the reasons of these results.

The result of first case is showed in Figs. 4 and 5.

(a) Comparison of R-square (b) Comparison of aMAE

Fig. 4. Predicting accuracy about "Lee Sedol v.s. AlphaGo"



(a) Comparison of R-square (b) Comparison of aMAE

Fig. 5. Predicting accuracy about "Brexit"

Figure 4 shows the experimental results on the dataset about the event "Lee Sedol v.s. AlphaGo". In the two figures, the CROP performs best if comparing with the baselines when $vis = 2$. If setting $vis = 2$, CROP increases the predicting accuracy by 8.9% in terms of the R-square and by 21.9% in terms of adjust Mean Absoluted Error. The results demonstrate that the predicting accuracy decreases when the vis increases from 2, with respect to both the R-square and the adjust Mean Absoluted Error. It is reasonable because the bigger vis means that more history data could be utilized which result in high variance.

The same results also happen at the experiments based on the event "Brexit". The Fig. 5 shows the experimental results. The results show that the predicting accuracy is decreasing when the value of *vis* increases, which is similar to the trend in the experiment of "Lee Sedo v.s. AlphaGo". Moreover, when the *vis* is setting as 1, the predicting accuracy is increased by 9.8% in terms of R-squre and increased by 6.2% in terms of adjust Mean Absoluted Error. Another similarity of the experimental result of these two datasets is that the perfomance of cro-SVR is better than SVR but worse than CROP, which shows that the cross-platform data has the potential to improve the predicting accuracy even though the matching alignment between the two time series is not considered.

In all, CROP has the best predicting performance and SVR has the worst predicting performance on the datasets of "Lee Sedol v.s. AlphaGo" and "Brexit". Morever, the small *vis* leads to the best performance. The results are reasonable, since the hot event "Lee Sedol v.s. AlphaGo" and "Brexit" are guided by a series of latest news. Thus, people's behaviors are strongly influenced by the recent information, which lead to the result that small *vis* is better.

The result of second case is showed in Fig. 6.

(a) Comparison of R-square

(b) Comparison of aMAE

Fig. 6. Predicting accuracy about "The Capsizing of a Big Ship"

The Fig. 6 shows the experimental result on the dataset about the event "The capsizing of a Big Ship". The result is that CROP and cro-SVR perform much better than SVR in terms of both R-square and adjust Mean Absolute Error. The R-square of SVR is approximately from 0 to 0.1 but that of co-SVR and CROP is approximately from 0.6 to 0.7. The adjust Mean Absolute Error of SVR is about from 0.6 to 0.7, but that of co-SVR and CROP is about 0.3 The performance difference may result from the property about the event. The event "The capsizing of a Big Ship" is started by the accident which shocked a plenty of Chinese so they maybe knew about the news without much details in the online social networks, and then search for the details on the search engine.

Thus, it is concluded that CROP performs best for cross-platform event popularity prediction model compared with other baseline methods and the cross-platform data can provide significantly improvement of prediction accuracy.

6 Conclusion

In this paper, we have proposed the CROP, an efficient cross-platform event popularity prediction model. CROP measures the popularity based on the TF-IDF method, denoises the data by Discrete Wavelet Transform method, explores the correlation of cross-platform event popularity and predicts the event popularity based on Support Vector Regression. Specially, CROP employs and improves the Dynamic Time Warping method to match two time series in different platforms and novelly define the aggregated popularity to establish the feature space of CROP, which could provide the inspiration for follow-up research. The real time dataset of hot events in China from 2015 to 2016 are used to validate the effectiveness of CROP.

Acknowledgement. This work is supported by the program of International S&T Cooperation (2016YFE0100300), the China 973 project (2014CB340303), the National Natural Science Foundation of China (Grant number 61472252, 61672353), the Shanghai Science and Technology Fund (Grant number 17510740200), and CCF-Tencent Open Research Fund (RAGR20170114).

References

1. Bandari, R., Asur, S., Huberman, B.A.: The pulse of news in social media: forecasting popularity. In: International AAAI Conference on Weblogs and Social Media (2012)
2. Chen, C., Xing, Z.: Towards correlating search on Google and asking on stack overflow. In: IEEE Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 83–92 (2016)
3. Gao, T., et al.: DancingLines: an analytical scheme to depict cross-platform event popularity. arXiv preprint arXiv:1712.08550 (2017)
4. Giraitis, L., Kokoszka, P., Leipus, R., Teyssière, G.: Rescaled variance and related tests for long memory in volatility and levels. J. Econ. **112**(2), 265–294 (2003)
5. Giummol, F., Orlando, S., Tolomei, G.: Trending topics on Twitter improve the prediction of Google hot queries. In: IEEE International Conference on Social Computing (SocialCom), pp. 39–44 (2013)
6. Giummolè, F., Orlando, S., Tolomei, G.: A study on microblog and search engine user behaviors: how Twitter trending topics help predict Google hot queries. Human **2**(3), 195 (2013)
7. Hoang, B.-T., Chelghoum, K., Kacem, I.: Modeling information diffusion via reputation estimation. In: Hartmann, S., Ma, H. (eds.) DEXA 2016. LNCS, vol. 9827, pp. 136–150. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44403-1_9
8. Keneshloo, Y., Wang, S., Han, E.H., Ramakrishnan, N.: Predicting the popularity of news articles. In: SIAM International Conference on Data Mining (ICDM), pp. 441–449 (2016)
9. Keogh, E.J., Pazzani, M.J.: Derivative dynamic time warping. In: SIAM International Conference on Data Mining (ICDM), pp. 1–11 (2001)
10. Leskovec, J., Backstrom, L., Kleinberg, J.: Meme-tracking and the dynamics of the news cycle. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 497–506 (2009)

11. Mathioudakis, M., Koudas, N.: TwitterMonitor: trend detection over the twitter stream. In: ACM SIGMOD International Conference on Management of Data (ICMD), pp. 1155–1158 (2010)
12. Miao, Z., et al.: Cost-effective online trending topic detection and popularity prediction in microblogging. ACM Trans. Inf. Syst. (TOIS) **35**(3), 1–36 (2016). Article no. 18
13. Rozenshtein, P., Anagnostopoulos, A., Gionis, A., Tatti, N.: Event detection in activity networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 1176–1185. ACM (2014)
14. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: ACM International Conference on World Wide Web (WWW), pp. 851–860 (2010)
15. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process. **26**(1), 43–49 (1978)
16. Schubert, E., Weiler, M., Kriegel, H.P.: SigniTrend: scalable detection of emerging topics in textual streams by hashed significance thresholds. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 871–880 (2014)
17. Shang, C., Panangadan, A., Prasanna, V.K.: Event extraction from unstructured text data. In: International Conference on Database and Expert Systems Applications (DEXA), pp. 543–557 (2015)
18. Struzik, Z.R., Siebes, A.: The Haar wavelet transform in the time series similarity paradigm. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 12–22. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-48247-5_2
19. Tang, Y., Ma, P., Kong, B., Ji, W., Gao, X., Peng, X.: ESAP: a novel approach for cross-platform event dissemination trend analysis between social network and search engine. In: Cellary, W., Mokbel, M.F., Wang, J., Wang, H., Zhou, R., Zhang, Y. (eds.) WISE 2016. LNCS, vol. 10041, pp. 489–504. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48740-3_36
20. Tolomei, G., Orlando, S., Ceccarelli, D., Lucchese, C.: Twitter anticipates bursts of requests for Wikipedia articles. In: ACM Workshop on Data-Driven User Behavioral Modelling and Mining from Social Media (DUBMOD), pp. 5–8 (2013)
21. Wang, S., Yan, Z., Hu, X., Philip, S.Y., Li, Z., Wang, B.: CPB: a classification-based approach for burst time prediction in cascades. Knowl. Inf. Syst. (KIS) **49**(1), 243–271 (2016)
22. Wang, S., Kam, K., Xiao, C., Bowen, S., Chaovalitwongse, W.A.: An efficient time series subsequence pattern mining and prediction framework with an application to respiratory motion prediction. In: AAAI Conference on Artificial Intelligence (AAAI) (2016)
23. Yang, J., Leskovec, J.: Patterns of temporal variation in online media. In: ACM International Conference on Web Search and Data Mining (WSDM), pp. 177–186 (2011)
24. Zheng, L., Jin, P., Zhao, J., Yue, L.: A fine-grained approach for extracting events on microblogs. In: Decker, H., Lhotská, L., Link, S., Spies, M., Wagner, R.R. (eds.) DEXA 2014. LNCS, vol. 8644, pp. 275–283. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10073-9_22

Probabilistic Classification of Skeleton Sequences

Jan Sedmidubsky(✉) and Pavel Zezula

Masaryk University, Brno, Czech Republic
{xsedmid,zezula}@fi.muni.cz

Abstract. Automatic classification of 3D skeleton sequences of human motions has applications in many domains, ranging from entertainment to medicine. The classification is a difficult problem as the motions belonging to the same class needn't be well segmented and can be performed by subjects of various body sizes in different styles and speeds. The state-of-the-art recognition approaches commonly solve this problem by training recurrent neural networks to learn the contextual dependency in both spatial and temporal domains. In this paper, we employ a distance-based similarity measure, based on deep convolutional features, to search for the k-nearest motions with respect to a query motion being classified. The retrieved neighbors are analyzed and re-ranked by additional measures that are automatically chosen for individual queries. The combination of deep features, dynamism in the similarity-measure selection, and a new kNN classifier brings the highest classification accuracy on a challenging dataset with 130 classes. Moreover, the proposed approach can promptly react to changing training data without any need for a retraining process.

1 Introduction and Related Work

Motion capture (MoCap) data (shortly *motion data*) are sequences of skeletons that consist of 3D positions of human body joints. These spatio-temporal data constitute a model, that on one hand substantially simplifies the view on the complex structure of human motion, but on the other hand enables automated and computer-aided processing. The increasing accuracy and availability of capturing devices have facilitated recording motion data in a variety of application domains, such as military, sports, medicine, law inforcement and smarthomes.

Such application domains require computer-aided operations to analyze the motion data automatically. One of the most fundamental operations is *action classification*, sometimes referred to as *action recognition*. It is the problem of inferring the kind of movement action, based on pre-classified training data. Solving this problem is challenging as the motions of the same kind can be performed by various subjects in different styles, speeds, and initial body postures.

The key part of the classification process is an efficient extraction of robust and sufficiently discriminative features to effectively model the spatial and temporal evolutions of different actions [15]. The survey in [10] categorizes existing

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 50–65, 2018.
https://doi.org/10.1007/978-3-319-98812-2_4

features into: (1) *joint-based* representations keeping the correlations among 3D joint locations within an action, (2) *mined-joint-based* features learning what subsets of body joints are more informative to discriminate a given action from the other ones, and (3) *dynamics-based* representations employing the advantage of the way the 3D joint locations move over time and modeling an action as a set of 3D trajectories, e.g., implemented by a 2D motion image approximating 3D joint positions by specific colors of the RGB color space [6,13].

The extracted features can then be compared for similarity by distance measures, such as the Dynamic Time Warping [2], to find the most similar training samples with respect to a query being classified. The retrieved samples are analyzed by k-nearest-neighbor classifiers to determine the class of the query [13]. On the other hand, the k-nearest neighbor (kNN) classifiers have been gradually effaced by the increasing success of deep neural networks. Several different neural-network architectures have recently been proposed for motion classification. Specifically, deep *convolutional* networks are trained by the 2D-motion-image features that are also classified by the network [6]. Most attempts suggest to employ the architecture of *recurrent* neural networks to better model the contextual dependency in the temporal domain [8]. This architecture can be enriched by the Long Short-Term Memory (LSTM) to better learn long-term temporal dependencies [9,15,18]. To further handle the noise and occlusion of skeleton sequences, *gating mechanisms* are integrated into LSTM to learn the reliability of the sequential data and accordingly adjust their effect on updating the long-term context information stored in the LSTM memory cell [8]. To benefit from different network architectures at the same time, the combination of convolutional and LSTM networks is proposed [12]. Recently, the recurrent neural networks have been enriched by *attention-based* mechanisms to additionally detect the most discriminative moments within an action [1,9,15].

Our Contribution

The state-of-the-art motion-data classifiers constitute either purposely-learned neural networks [1,6,9], or 1NN classifiers dependent on a single similarity measure [13,14]. Even if the machine-learning classifiers generally achieve a higher accuracy, they have to be retrained each time when new classes or even only class samples are added, which is a very time-consuming process making the real-time processing prohibited. We plan to overcome this problem by proposing new k-nearest-neighbor classifiers that can dynamically react to changing training data, while benefiting from the similarity concept originally learned on a deep convolutional neural network. In particular, we extend the 1NN approach by introducing two new kNN classifiers that additionally output a probability distribution over classes to which a query motion should belong. More importantly, we propose a third *confusion-based* classifier that employs the kNN-based probability distribution to automatically select more convenient similarity measures for classifying the query motion, rather than relying only on some global similarity. Despite reaching the best accuracy on a challenging dataset with 130

classes, the proposed approach does not require large volumes of training data and can react to data changes immediately in contrast to the recent approaches.

2 Representation and Similarity of Skeleton Sequences

A *motion sequence* (or simply *motion*) M is represented by a sequence of consequent *skeletons*. The total number $|M|$ of skeletons determines the motion *length*. The t-th skeleton $\in \mathbb{R}^{31\times 3}$ captured at the time moment t $(1 \leq t \leq |M|)$ consists of 3D coordinates of 31 tracked *joints*.

To compare a pair of skeleton sequences, we adopt the state-of-the-art similarity measure which was originally proposed in [14] and improved in [13]. It uses the Euclidean distance to compare 4,096-dimensional *feature vectors* extracted from motions of variable lengths. A feature vector is extracted from a motion sequence M in the following three steps.

1. *Normalization* – Each skeleton is normalized to become independent of both position and orientation in a 3D space. The skeleton size is additionally scaled to keep the same body proportions over all motions.
2. *Motion-image construction* – The 3D space covering all possible normalized skeletons is then mapped into the RGB color space to approximate any joint position by specific color. The positions of all joints changing over time are projected into a 2D image, so-called *motion image*. This image is generated by the "WeightedJointsFixed" variant [13] to occupy $256 \times |M|$ pixels.
3. *Feature extraction* – The generated motion image is finally processed by the *reference model*[1] of the well-known Krizhevsky's deep convolutional neural network [5] to extract a deep 4,096-dimensional feature vector F_M^{4K}, as the output of the last hidden network layer.

To achieve a higher descriptive power of feature vectors, the reference model originally trained on common photographs is additionally *fine-tuned* by a training set of motion images, as described in the experiments. The feature extraction and fine-tuning processes are deeply analyzed in [13,14].

The extracted 4,096D feature vectors $F_{M_1}^{4K}$ and $F_{M_2}^{4K}$ of motions M_1 and M_2 are compared by the Euclidean distance as:

$$compDist^{4K}\left(F_{M_1}^{4K}, F_{M_2}^{4K}\right) = \left\|F_{M_1}^{4K} - F_{M_2}^{4K}\right\|.$$

We call this similarity measure as *original* and employ it by all the classifiers proposed in this paper.

3 Classification Baseline

We formally define the problem of motion classification and introduce the baseline 1NN classification approach, which has already been evaluated on the motion-image similarity measure in [13].

[1] https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet.

Classification is the problem of identifying a single *class* (sometimes referred to as *category*) to which a given motion sequence belongs, based on a set of already categorized motion sequences, i.e., *training data*. Formally, a general classifier $classify$ determining the class of a *query* motion is defined as:

$$classify : \mathcal{M} \to \mathcal{C}, \tag{1}$$

where $\mathcal{M} = \{M_1, \ldots, M_n\}$ denotes the input domain of n motions, while $\mathcal{C} = \{C_1, \ldots, C_m\}$ is the output domain of m classes.

3.1 1NN Classifier

To implement any search-based classifier, there is a need to retrieve the most similar motions – *nearest neighbors* (NN) – with respect to a query motion. The following $1NN$ function searches a training set $\mathcal{M}^{\mathrm{TR}}$ of categorized motion sequences and returns the one that is the most similar to the query motion M^{Q}:

$$\begin{aligned} 1NN\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}\right) = \Big\{ & M' \in \mathcal{M}^{\mathrm{TR}} \mid \forall M \in \mathcal{M}^{\mathrm{TR}} : \\ & compDist^{4\mathrm{K}}\left(F^{4\mathrm{K}}_{M^{\mathrm{Q}}}, F^{4\mathrm{K}}_{M'}\right) \leq compDist^{4\mathrm{K}}\left(F^{4\mathrm{K}}_{M^{\mathrm{Q}}}, F^{4\mathrm{K}}_{M}\right) \Big\}. \end{aligned} \tag{2}$$

The 1-*nearest-neighbor classifier*, denoted as $classify^{1\mathrm{NN}}$, then simply assigns the query motion the class of the most similar motion:

$$classify^{1\mathrm{NN}}\left(M^{\mathrm{Q}}\right) = getClass\left(1NN\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}\right)\right), \tag{3}$$

where the function $getClass$ returns the known class $C_i \in \mathcal{C}\,(i \in \{1, \ldots, m\})$ of motion passed in the argument.

3.2 Experimental Evaluation

We first introduce the dataset and methodology that are used for evaluation throughout this paper. Then we present the accuracy results of the 1NN classifier.

Dataset. The classification scenario is evaluated on 3D skeletal data of the publicly available HDM05 [11] motion capture dataset. This dataset provides the ground truth HDM05-130, which categorizes 2,345 short motion sequences into 130 categories of specific actions, such as the turn left, sit down on a chair, or clap with hands five times. The average action length is 2.17 s – it corresponds to about 260 frames with the dataset sampling frequency of 120 Hz.

We select this HDM05-130 ground truth because it is very challenging – it contains the highest number of 130 categories when compared to other datasets, such as CMU (30 categories), NTU RGB + D (60 categories), MSR (20 categories) or MHAD (11 categories) [16]. Moreover, it is characterized with a subtle categorization, containing for example separate classes for kicking with left or right leg, to the front or to the side.

As suggested in [13,14], we additionally ignore 8 categories with less than 10 motion samples. Thus our resulting ground truth HDM05-122 contains 122 categories with 2,328 motions in total.

Methodology. To be consistent with previous works [6,7,13], we use 50% of motion sequences for training by applying the 2-fold cross validation procedure. In particular, we *randomly* partition all 2,328 motion samples into halves, i.e., into two sets (folds) of 1,164 motions. In the first pass, the first fold is used for training ($\mathcal{M}^{\mathrm{TR}}$) and the second one for testing. In the second pass, the training and test folds are swapped. For each pass, we preprocess the ground truth to:

1. Generate motion images (see Sect. 2) for both training and test motions;
2. Fine-tune the reference model of the neural network by the motion images from the training set;
3. Extract a 4,096D feature vector for each of training and test images based on the fine-tuned network model.

The test motions are then used as query arguments of the $classify^{\mathrm{1NN}}$ classifier. The accuracy of a single-motion classification is either 100%, or 0% based on the fact whether the classified category equals to the category of the query motion. The accuracy of the given pass is then measured as an average accuracy over 1,164 test queries. The accuracy of the classifier is finally expressed as an average over both passes.

Evaluation of Classification Accuracy. As already demonstrated in [13], the 4,096D deep features compared by the Euclidean distance achieves the 1NN classification accuracy of 87.84%. This result serves as the baseline for other classifiers proposed in this paper.

4 Probabilistic *k*NN Classifiers

Although the 1NN classifier is simple and quite accurate, it needn't be convenient when (1) the query-closest neighbors have almost the same distance while belonging to different classes or when (2) the correct class is confusable with another class, e.g., "grab a thing" with"deposit a thing". In that cases, it is useful to analyze the categories and similarities of more nearest neighbors, rather than relying only on the most similar one. This additionally brings the possibility to determine a probability distribution over a set of classes that the query should belong to. Formally, a *probabilistic classifier* is defined as:

$$classify : \mathcal{M} \rightarrow \{(\mathcal{C} \times [0,1])\}\,. \tag{4}$$

The result of classification is a set of pairs associating the classes with their probabilities of being the correct match. Within this section, we propose two probabilistic classifiers $classify^{\mathrm{kNN}}$ and $classify^{\mathrm{kNN\_TMC}}$.

4.1 Weighted-Distance *k*NN Classifier

The idea is to classify the query by a majority vote of its nearest neighbors. We consider not only the number of votes but also the similarity of neighbors with respect to the query.

To obtain the k-nearest neighbors from the categorized training set $\mathcal{M}^{\mathrm{TR}}$ to the query M^{Q}, the following kNN function is evaluated:

$$\begin{aligned} kNN\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right) = \Big\{ & M' \in \mathcal{M}' \;\Big|\; \mathcal{M}' \subset \mathcal{M}^{\mathrm{TR}}, |\mathcal{M}'| = k, \\ & \forall M' \in \mathcal{M}', \forall M \in \mathcal{M}^{\mathrm{TR}} / \mathcal{M}' : \\ & compDist^{4\mathrm{K}}\left(F^{4\mathrm{K}}_{M^{\mathrm{Q}}}, F^{4\mathrm{K}}_{M'}\right) \leq compDist^{4\mathrm{K}}\left(F^{4\mathrm{K}}_{M^{\mathrm{Q}}}, F^{4\mathrm{K}}_{M}\right) \Big\}. \end{aligned} \tag{5}$$

We extend this kNN function by the additional class parameter $C \in \mathcal{C}$ to determine the subset of nearest neighbors that belong to the specified class C:

$$kNN(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k, C) = \left\{ M \in kNN(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k) : getClass(M) = C \right\}. \tag{6}$$

Relevance of Neighbors. As the k-nearest neighbors are retrieved, the *relevance* of each neighbor for classification is determined. This relevance is computed by normalizing the neighbor distance with respect to the distance of the k-th neighbor. Such query-dependent normalization is very effective in situations where distances of nearest neighbors vary a lot across different categories, rather than a normalization based on the maximum possible distance. The relevance of the neighbor M is computed by the function $compRel : \mathcal{M} \rightarrow [0, 1]$ as:

$$\begin{aligned} compRel(M) &= 1 - \frac{compDist^{4\mathrm{K}}(F^{4\mathrm{K}}_{M^{\mathrm{Q}}}, F^{4\mathrm{K}}_{M})}{kNeighborDist \cdot kNeighborDistWeight}, \\ kNeighborDist &= \max\left\{ compDist^{4\mathrm{K}}(F^{4\mathrm{K}}_{M^{\mathrm{Q}}}, F^{4\mathrm{K}}_{M}) : M \in kNN\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right) \right\}, \end{aligned} \tag{7}$$

where $kNeighborDist$ is the precomputed distance to the k-th neighbor and $kNeighborDistWeight \in [1, \infty)$ is a user-defined parameter determining the importance of distances to classification. We fix this parameter to 1.1 to put a high emphasis on the distances, which at the same time decreases relevance of the k-th neighbor a lot.

The result relevance approaching the value 1 denotes a high importance of the specific neighbor and with a decreasing value, the neighbor importance goes down.

Aggregation of Relevances of Neighbors. To compute classification probabilities, relevances of neighbors belonging to the same class are summed and finally normalized across all categories. The following *compRels* function returns the pairs associating the summed relevance r_i with the class C_i:

$$\begin{aligned} compRels\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right) = \Big\{ & (C_i, r_i) \;\Big|\; C_i \in \mathcal{C}, \\ & r_i = \sum_{M \in kNN(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k, C_i)} compRel(M) \Big\}. \end{aligned} \tag{8}$$

To normalize the relevances into probabilities, the following *normRels* function is employed:

$$\begin{aligned} &normRels\left(\{(C_1, r_1), \ldots, (C_m, r_m)\}\right) = \\ &\left\{ \left(C_i, p_i\right) \;\middle|\; i \in [1, m], p_i = r_i / \textstyle\sum_{j=1}^{m} r_j \right\}. \end{aligned} \tag{9}$$

It transforms the relevance r_i of each class C_i into probability $p_i \in [0, 1]$, computed as a ratio between the class relevance and the sum of relevances of all the classes. Such normalization additionally ensures that the sum of probabilities of all the classes is equal to 1. The classifier returning such probabilities of classes is denoted as *weighted-distance classifier* ($classify^{\mathrm{kNN}}$) and defined as:

$$classify^{\mathrm{kNN}}\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right) = normRels\left(compRels\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right)\right). \tag{10}$$

4.2 Training-Class-Sizes *k*NN Classifier

The disadvantage of the previous weighted-distance classifier is that the computed class probabilities can be influenced by different sizes of categories, in terms of the number of training samples. For example, assume that the query M^{Q} should be assigned to the class C_1 and that the classes C_1 and C_2 contain 1 and 100 training samples, respectively. Then by considering $k = 15$, the 15 nearest neighbors to M^{Q} are retrieved. Even if the most similar neighbor belongs to the correct class C_1, the class C_2 is evaluated as the most probable because next 14 neighbors belonging to C_2 overweight just one sample. This can arise when the sizes of training classes are not balanced.

In order to avoid such situation, we compute for each class the ratio between the number of its samples being among the k-nearest neighbors and the number of the available training samples of that class. The function *updRels* updates the originally-computed relevances r_i by the square root of such ratios in order to calculate new relevances r_i' as:

$$\begin{aligned} updRels\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right) = \Big\{ &\left(C_i, r_i'\right) \;\Big|\; \left(C_i, r_i\right) \in compRels\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right), \\ &r_i' = r_i \cdot \sqrt{\frac{|kNN(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k, C_i)|}{|\{M \in \mathcal{M}^{\mathrm{TR}} : getClass(M) = C_i\}|}} \Big\}. \end{aligned} \tag{11}$$

The classifier considering the sizes of classes is denoted as *training-class-sizes classifier* ($classify^{\mathrm{kNN\_TCS}}$) and formally defined as:

$$classify^{\mathrm{kNN\_TCS}}\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right) = normRels\left(updRels\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right)\right). \tag{12}$$

4.3 Evaluation of Classification Accuracy

Both the proposed kNN classifiers ($classify^{\mathrm{kNN}}$ and $classify^{\mathrm{kNN\_TCS}}$) compute the probabilities of classes of being the correct match. To evaluate the classification scenario, we consider the class of the highest probability. Figure 1a illustrates the differences between the classifiers depending on an increasing value of

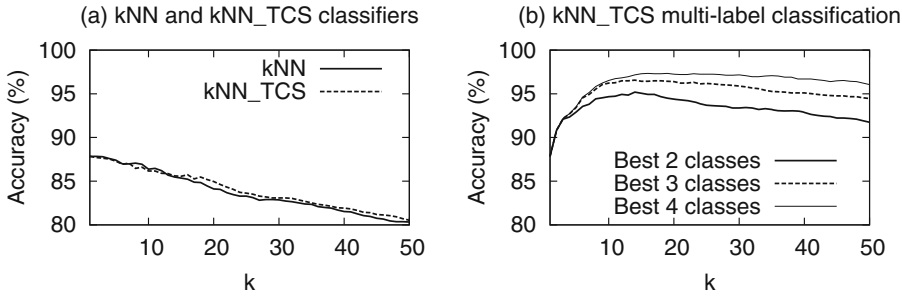


Fig. 1. Accuracy of kNN classification based on a varying value of k: (a) Accuracy of $classify^{\text{kNN}}$ and $classify^{\text{kNN\_TCS}}$ classifiers. (b) Accuracy of the $classify^{\text{kNN\_TCS}}$ classifier when the correct match is among the two/three/four highest-probable classes.

k. As expected, the differences are relatively small due to a quite balanced sizes of training classes. In particular, the differences are negligible up to the value $k = 15$, which is the number approaching the average number of class samples. With the value $k > 15$, the sizes of training classes play a more important role, which leads to a slightly better accuracy of the $classify^{\text{kNN\_TCS}}$ classifier.

Both classifiers reach the highest accuracy when $k = 1$ and with an increasing value, their accuracy decreases. Even the results do not outperform the baseline 1NN classifier (the accuracy of 87.84%), the probabilities of individual classes bring new possibilities for enhancements. Based on these kNN classifiers, the confusion-based kNN classifier is proposed in next section.

5 Confusion-Based Classifier

In case there is no clear decision on what class is a query motion, the kNN classifiers provide more classes ranked by their probabilities of being the correct match. If we tolerate that the correct match is among the two top ranked classes (instead of considering only the highest-probable class), the classification accuracy can significantly increase. Figure 1b illustrates the kNN_TCS classification accuracies when the correct match is included in the two/three/four top ranked classes. We can see that for $k = 15$, the achieved accuracy is higher than 95%, even when only the two top ranked classes are considered.

To significantly improve the accuracy from the baseline value of 87.84% up to 95%, we would need a magic stick choosing the correct class from the two top ranked classes obtained by the 15NN_TCS classifier. The idea of approximating the magic stick is to *re-rank* the k-nearest neighbors based on different similarity measures which can better separate the two top ranked classes, than the original measure presented in Sect. 2. Such suitable measures are automatically selected for each pair of classes based on *confusion matrices* learned from the training data. The class of the neighbor with the smallest re-ranked distance is finally considered as the classification result.

5.1 Training Phase: Learning Confusion Matrices

To increase the classification accuracy, additional distance measures can be provided. We consider the set D of additional descriptors $\{D_1, \ldots, D_d\}$ to represent each motion M by additional feature vectors $F_M^{D_1}, \ldots, F_M^{D_d}$. For the i-th descriptor D_i ($i \in [1, d]$), the similarity of feature vectors $F_{M_1}^{D_i}$ and $F_{M_2}^{D_i}$ of motions M_1 and M_2 is calculated using a predefined distance measure, denoted as $compDist^{D_i}$.

Within the training phase, the *confusion matrix* cm^{D_i} is calculated for each of the measures. Each matrix has the size of $m \times m$ elements, where m is the number of training classes. The element $cm^{D_i}[C_x, C_y] \in [0, 1]$ of the matrix expresses how much the class C_x is *confusable* with the class C_y ($x, y \in [1, m]$), with respect to the distance measure $compDist^{D_i}$. The value of 0 means that the measure perfectly separates the training motions of both classes and with an increasing value, separability decreases. We compute such value by evaluating the average 1NN classification accuracy over all training motions categorized in both classes. Formally, we compute the value as:

$$\begin{aligned} cm^{D_i}[C_x, C_y] &= \frac{\left|\left\{M \in \mathcal{M}^{C_x} \mid getClass(M) \neq classify^{\mathrm{1NN}}(M, \mathcal{M}^{C_{xy}}/\{M\})\right\}\right|}{|\mathcal{M}^{C_x}|}, \\ \mathcal{M}^{C_x} &= \left\{M \in \mathcal{M}^{\mathrm{TR}} \mid getClass(M) = C_x\right\}, \\ \mathcal{M}^{C_{xy}} &= \left\{M \in \mathcal{M}^{\mathrm{TR}} \mid getClass(M) \in \{C_x, C_y\}\right\}, \end{aligned} \tag{13}$$

where $\mathcal{M}^{C_x}$ denotes the training motions belonging to the class C_x, while $\mathcal{M}^{C_{xy}}$ represents motions of both classes C_x and C_y.

The way of matrix calculation does not guarantee the symmetry, i.e., it may happen that $cm^{D_i}[C_x, C_y] \neq cm^{D_i}[C_y, C_x]$. Since this is not convenient for further processing, we make the matrix symmetric by considering the maximum value, which can increase the confusion value of both classes:

$$cm^{D_i}[C_x, C_y] = cm^{D_i}[C_y, C_x] = \max\left\{cm^{D_i}[C_x, C_y], cm^{D_i}[C_y, C_x]\right\}. \tag{14}$$

5.2 Classification Phase: Analyzing the Nearest Neighbors

Within the classification phase, the k-nearest neighbors are retrieved and classified by the $classify^{\mathrm{kNN\_TCS}}$ classifier. The correct match is considered to be among the two top ranked classes. Based on the confusion values of these two classes, the weights of the additionally provided similarity measures are computed. These weights are used to re-rank the nearest neighbors. The whole process is described in the following paragraphs.

Identifying the Most Ranked Classes. The two top ranked classes C' and C'' ($C', C'' \in \mathcal{C}$) are selected from the $classify^{\text{kNN\_TCS}}$ classification result as:

$$\begin{aligned} get2MostRankedClasses\left(M^{\text{Q}}, \mathcal{M}^{\text{TR}}, k\right) = & \Big\{ C', C'' \;\Big| \\ (C', p'), (C'', p'') \in classify^{\text{kNN\_TCS}}(M^{\text{Q}}, \mathcal{M}^{\text{TR}}, k) : & \; p' \geq p'', \\ \forall (C_i, p_i) \in classify^{\text{kNN\_TCS}}(M^{\text{Q}}, \mathcal{M}^{\text{TR}}, k) / \{(C', p')\} : & \; p'' \geq p_i \Big\}. \end{aligned} \tag{15}$$

Weighting Similarity Measures. The most suitable similarity measure(s) for re-ranking should have the lowest (ideally zero) confusion value for the two top ranked classes C' and C'' – the lower the value $cm^{D_i}[C', C'']$, the better separation ability of the D_i descriptor. Such lowest confusion value, denoted as $minConf$, can be easily obtained by this formula: $minConf = \min_{i=1}^{d} cm^{D_i}[C', C'']$.

The lowest confusion value is used to determine the *weight* $w^{D_i} \in [0, 1]$ for each descriptor D_i. Since there can be a high number of provided descriptors, we select only the best one(s) of the lowest confusion value in order not to suppress important movement features by aggregating many measures. At the same time, the best-available descriptor(s) for the query motion needn't be still optimal and can have a quite low weight, i.e., the two top ranked classes cannot be simply separated by any of the additionally provided measures. For these reasons, we decrease the weight to the power of 3 to more suppress the influence of the best-available measure(s) having "lower" weights. The descriptor weight w^{D_i} is finally determined as:

$$w^{D_i} = \begin{cases} 0 & cm^{D_i}[C', C''] > minConf, \\ (1 - minConf)^3 & cm^{D_i}[C', C''] = minConf. \end{cases} \tag{16}$$

If the best-available measure(s) have a "low" weight, the original motion-image similarity measure is more important. The weight $w^{4\text{K}}$ of such original measure is determined as:

$$w^{4\text{K}} = \max\left\{ \left(1 - cm^{4\text{K}}[C', C'']\right)^3, 1 - (1 - minConf)^3 \right\}. \tag{17}$$

This value is the maximum between the weight computed based on the confusion matrix of the original descriptor and the inverse value of the weight of the best-available additional measure. In other words, the original similarity measure is preferred if it well separates the two top ranked classes or in cases when a strong additional descriptor is not available.

Re-ranking and Classifying Neighbors. The calculated weights of the original similarity measure and additional measures are used to re-rank the list of the retrieved neighbors. The re-ranking process is based on weighting and normalizing the distances separately for each descriptor and summing such updated

distances together. Formally, to re-rank the neighbor M with respect to the query M^{Q}, the following $reRank$ function is applied:

$$reRank(M^{\mathrm{Q}}, M) = w^{4K} \cdot compDist^{4\mathrm{K}}(F^{4\mathrm{K}}_{M^{\mathrm{Q}}}, F^{4\mathrm{K}}_{M})/md^{4\mathrm{K}}[C', C''] + \sum_{i=1}^{d} \left(w^{D_i} \cdot compDist^{D_i}(F^{D_i}_{M^{\mathrm{Q}}}, F^{D_i}_{M})/md^{D_i}[C', C''] \right), \quad (18)$$

where md^{D_i} is the matrix of class-pairwise maximum distances. In particular, the value $md^{D_i}[C', C'']$ is the maximum of distances computed among all pairs of training motions belonging to classes C' or C'', with respect to the distance measure $compDist^{D_i}$. Such matrices are simply precomputed for the original motion-image measure (matrix $md^{4\mathrm{K}}$) and each additional measure (matrices md^{D_i}) in the training phase as:

$$md^{D_i}[C', C''] = \max \left\{ compDist^{D^i}(F^{D_i}_{M_1}, F^{D_i}_{M_2}) \mid M_1 \in C', M_2 \in C'' \right\}. \quad (19)$$

The maximum distances computed separately for each pair of classes more effectively normalize the distances, rather than considering a global maximum. The neighbors are than sorted based on their re-ranked distance and the class of that with the lowest distance is considered as the final classification result. We call this approach as the *confusion-based* classifier $classify^{\mathrm{CONF}}$ and define it formally as:

$$classify^{\mathrm{CONF}}\left(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k\right) = getClass\left(\{\ M' \in kNN(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k) \mid \forall M \in kNN(M^{\mathrm{Q}}, \mathcal{M}^{\mathrm{TR}}, k) : reRank\left(M^{\mathrm{Q}}, M'\right) \leq reRank\left(M^{\mathrm{Q}}, M\right)\ \}\right). \quad (20)$$

Due to the concentration on the first re-ranked neighbor only, the confusion-based classifier does not support probabilistic classification. However, it would be possible if the re-ranked list was processed by another kNN classifier.

5.3 Experimental Evaluation

We describe the additional three measures used for re-ranking, evaluate the accuracy of the confusion-based classifier, and outline the efficiency results.

Description of Additional Similarity Measures. To verify the suitability of the confusion-based classifier, we implement the following three simple descriptors whose features are compared by the Manhattan distance, i.e., L_1 metric.

- *Joint trajectory length* (D_1) – the 31D feature vector, where each dimension corresponds to the total length of the trajectory of the specific joint (out of 31 joints). The trajectory length is computed by summing the Euclidean distances between the 3D joint positions in consecutive skeletons.

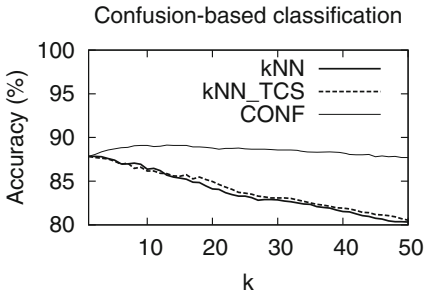


Fig. 2. Accuracy between kNN-based and confusion-based classifiers.

- *Normalized joint trajectory length* (D_2) – the 31D feature vector constructed in the same way as the previous D_1 descriptor. The total trajectory lengths are additionally normalized by the length of the motion sequence, i.e., by the number of captured skeletons.
- *Maximum axis distance* (D_3) – the 93D feature vector whose dimensions correspond to the maximum reachable coordinate separately in the $x/y/z$ axis of each joint. When comparing with the feature of another motion, only the differences in the 8 most distant dimensions are considered, which implies that this similarity measure is not symmetric.

While the first two descriptors (D_1 and D_2) are computed based on the original motion data, the third descriptor (D_3) is extracted from normalized skeleton-centric view-invariant data, described in Sect. 2.

Classification Accuracy. Figure 2 illustrates the contrast between the accuracies of both kNN classifiers (introduced in Sect. 4) and the confusion-based classifier. The confusion-based classifier increases the accuracy with an increasing value of k up to $k = 15$ and then the accuracy slightly decreases. When $k = 15$, the classification reaches the 89.09% accuracy. When compared to the baseline 1NN accuracy of 87.84%, the error in classification is decreased by 10%.

Efficiency of Learning and Classification. To pre-compute a confusion matrix cm^{D_i} $(m \times m)$ for a given descriptor D_i, there is a need to evaluate m^2 confusion values, where m denotes the number of provided classes. Assuming that all the classes contain approximately the same number of $|\mathcal{M}^{\mathrm{TR}}|/m$ training motion samples, a single matrix cell requires $((|\mathcal{M}^{\mathrm{TR}}| - 1)/m) \cdot |\mathcal{M}^{\mathrm{TR}}|/m$ evaluations of the D_i distance measure. In total, the following number of:

$$m^2 \cdot \left(\frac{|\mathcal{M}^{\mathrm{TR}}| - 1}{m} \cdot \frac{|\mathcal{M}^{\mathrm{TR}}|}{m} \right) \cong |\mathcal{M}^{\mathrm{TR}}|^2$$

$compDist^{D_i}$ distance computations is performed in the training phase. Since in our case $|\mathcal{M}^{\mathrm{TR}}| = 1{,}164$, we perform about 1.4 M distance computations for the

original as well as each of the three additional descriptors. The total time needed to learn all the matrices takes only about 10 s. For example, the clearly most expensive $compDist^{4K}$ measure applied to the 4,096D features needs roughly 7 s on commodity hardware (i7 960 at 3.2 GHz).

The md^{D_i} matrix keeping the maximum class-pairwise distances can be computed during the process of calculation of the confusion matrix.

Also the classification time is not much influenced by additional descriptors that need roughly 1 ms in total for re-ranking. The most expensive operation is the searching for k-nearest neighbors, which requires 6 ms. However, this time can be further decreased by indexing the original features by any metric-based structure [17] to scale to large databases of training samples. Sometimes, the feature extraction process is included in classification time, mainly when queries cannot be preprocessed in advance. In our case, the extraction of all the features, also including the original 4,096D feature, takes about 30 ms per a query motion.

6 Comparison with the State-of-the-Art Approaches

We compare the accuracy of the best-performing confusion-based classifier with the most recent approaches [4,6,7,13,14] that evaluate the same 2-fold cross validation procedure on the challenging HDM05 ground truth with 122 or 130 categories. Table 1 shows that we beat not only the 1NN classifiers based on the motion-image concept [13,14] but even the most recent purposely-trained classifiers [4,6,7] based on neural networks. From the recognition-error point of view, we decrease the error by 54%, 33%, 15%, 10%, and 19% with respect to the methods in [4,6,7,13,14], respectively.

Table 1. Comparison with the state-of-the-art methods using the 2-fold cross validation (i.e., using 50% of training data).

Method	Accuracy (%)	
	HDM05-122	HDM05-130
Huang et al. [4]	N/A	75.78
Laraba et al. [6]	N/A	83.33
Sedmidubsky et al. [14]	87.24	N/A
Sedmidubsky et al. [13]	87.84	N/A
Li et al. [7]	N/A	86.17
Our approach	**89.09**	**88.78**

Although there are other well-performing classifiers, such as [1,3,9,12,15], they do not evaluate the recognition accuracy on the challenging set of 122 or 130 HDM05 categories. One of the reasons is probably a limited amount of training data, with less than twenty samples per a single class. Moreover, such

approaches pay a little attention to efficiency and applicability issues. Specifically, our confusion-based classifier has the following advantages when compared to the state-of-the-art classifiers.

- *Dynamic measure selection* – additional similarity measures are dynamically and automatically weighted to select the best possible one(s) for re-ranking, with respect to a query motion being classified. It is much more convenient than training a single descriptor that needs to recognize any class, as used in existing works.
- *Robustness* – if there are some misclassified training samples, they can degrade the classification accuracy. In our approach, the class of such samples can simply be repaired causing the immediate impact on classification, without the necessity of any long-term retraining like in [9,12].
- *Limited amount of training data* – the original descriptor does not need large amounts of training data due to the utilization of pre-trained neural network on a different image domain. Additional measures can also separate well classes having a limited number of training samples. In particular, we use 50% of the dataset for training in comparison with other approaches utilizing, e.g., 80% or 90% [14] of data for training.
- *Efficiency and indexability* – the comparison of the original 4,096D feature vectors by the Euclidean distance enables indexing such features by any metric-based structure [17] to efficiently evaluate k-nearest neighbor queries, even in very large databases of training samples.

7 Conclusions

The state-of-the-art similarity measure for motion data [13,14], based on 4,096D deep features extracted using a convolutional neural network, achieves a high accuracy even when used with the baseline 1NN classifier. In this paper, we employ this original measure to search for the k-nearest training samples with respect to an unlabeled query. The retrieved neighbors are analyzed by the proposed kNN_TCS classifier to determine the two top probable classes for the processed query. To select the correct class, we employ additional simple similarity measures. Based on class-pairwise confusion matrices automatically learned for each similarity measure, the correct class is selected in 94% of cases. This helps decrease the error in classification by 10% on the challenging HDM05-122 dataset, when compared to the baseline 1NN classifier.

The proposed approach, despite being very effective in classification, has several other advantages in comparison with existing classifiers. In particular, the training sample set can be (1) *small* (less than 10 samples per class) to reach a high classification accuracy or (2) *large* to search the k-nearest neighbors efficiently due to indexability of the original similarity measure, and (3) *dynamic* (even in sense of adding samples of new classes) with the immediate impact to classification, without any time-consuming retraining process. Moreover, the additional similarity measures are dynamically utilized in classification

only when the original measure does not provide clear results. The provided additional measures are automatically integrated into the confusion-based classifier, without any need of supervision.

Acknowledgment. This research was supported by GBP103/12/G084.

References

1. Baradel, F., Wolf, C., Mille, J.: Human action recognition: pose-based attention draws focus to hands. In: ICCV Workshop on Hands in Action, Venice, Italy (2017)
2. Barnachon, M., Bouakaz, S., Boufama, B., Guillou, E.: Ongoing human action recognition with motion capture. Pattern Recogn. **47**(1), 238–247 (2014)
3. Bütepage, J., Black, M.J., Kragic, D., Kjellström, H.: Deep representation learning for human motion prediction and classification. CoRR abs/1702.07486 (2017)
4. Huang, Z., Wan, C., Probst, T., Gool, L.V.: Deep learning on lie groups for skeleton-based action recognition. CoRR abs/1612.05877 (2016)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105. Curran Associates, Inc. (2012)
6. Laraba, S., Brahimi, M., Tilmanne, J., Dutoit, T.: 3D skeleton-based action recognition by representing motion capture sequences as 2D-RGB images. Comput. Anim. Virtual Worlds **28**(3–4), e1782 (2017)
7. Li, C., Cui, Z., Zheng, W., Xu, C., Yang, J.: Spatio-temporal graph convolution for skeleton based action recognition. In: 32nd Conference on Artificial Intelligence (AAAI 2018). AAAI Press (2018)
8. Liu, J., Shahroudy, A., Xu, D., Wang, G.: Spatio-temporal LSTM with trust gates for 3D human action recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 816–833. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_50
9. Liu, J., Wang, G., Duan, L., Hu, P., Kot, A.C.: Skeleton based human action recognition with global context-aware attention LSTM networks. IEEE Trans. Image Process. **27**(4), 1586–1599 (2018)
10. Lo Presti, L., La Cascia, M.: 3D skeleton-based human action classification. Pattern Recognit. **53**(C), 130–147 (2016)
11. Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., Weber, A.: Documentation Mocap Database HDM05. Technical report CG-2007-2, Universität Bonn (2007)
12. Nez, J.C., Cabido, R., Pantrigo, J.J., Montemayor, A.S., Vlez, J.F.: Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. Pattern Recognit. **76**, 80–94 (2018)
13. Sedmidubsky, J., Elias, P., Zezula, P.: Enhancing effectiveness of descriptors for searching and recognition in motion capture data. In: 19th International Symposium on Multimedia, pp. 240–243. IEEE Computer Society (2017)
14. Sedmidubsky, J., Elias, P., Zezula, P.: Effective and efficient similarity searching in motion capture data. Multimed. Tools Appl. **77**(10), 12073–12094 (2018). https://doi.org/10.1007/s11042-017-4859-7
15. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An End-to-End spatio-temporal attention model for human action recognition from skeleton data. CoRR abs/1611.06067 (2016). http://arxiv.org/abs/1611.06067

16. Wang, P., Li, W., Ogunbona, P.O., Wan, J., Escalera, S.: RGB-D-based motion recognition with deep learning: a survey. Int. J. Comput. Vis. **PP**(99), 1–34 (2017)
17. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach. Advances in Database Systems, vol. 32. Springer, Boston (2006). https://doi.org/10.1007/0-387-29151-2
18. Zhu, W., et al.: Co-Occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In: 30th Conference on Artificial Intelligence (AAAI 2016), pp. 3697–3703. AAAI Press (2016)

Uncertain Information

A Fuzzy Unified Framework for Imprecise Knowledge

Soumaya Moussa[1(✉)] and Saoussen Bel Hadj Kacem[1,2]

[1] COSMOS, National School of Computer Sciences, University of Manouba, Manouba, Tunisia
{soumaya.moussa, saoussen.belhadjkacem}@ensi-uma.tn
[2] Faculty of Economic Sciences and Management of Nabeul, University of Carthage, Nabeul, Tunisia

Abstract. When building Knowledge-Based Systems, we are often faced with vague data. The formers are generally modeled and treated using fuzzy logic, which is based on fuzzy set theory, or using symbolic multi-valued logic, which is based on multi-set theory. To provide a unified framework to handle simultaneously both types of information, we propose in this paper a new approach to translate multi-valued knowledge into fuzzy knowledge. For that purpose, we put forward a *symbolic-to-fuzzy conversion* method to automatically generate fuzzy sets from an initial multi-set. Once unified, handling heterogeneous knowledge become feasible. We apply our proposal in Rule-Based Systems where an approximate reasoning is required in their inference engine. Once new facts are deduced and in order to make the translation completely transparent for the user, we also provide a *fuzzy-to-symbolic conversion* method. Its purpose is to restore the original knowledge type if they were multi-valued. Our proposal offer a high flexibility to the user to reason regardless to the knowledge type. In addition, it is an alternative to overcome the modeling shortcoming of abstract data by taking advantage of a rigorous mathematical framework of fuzzy logic. A numerical study is finally provided to illustrate the potential application of the proposed methodology.

Keywords: Fuzzy logic · Multi-valued logic · Vagueness · Unified framework Knowledge-Based Systems

1 Introduction

Because of the variety of information sources, the lack of reliability of the measurement tools and the subjectivity of the information evaluation that differ from an expert to another, knowledge are generally imperfect and heterogeneous. Hence, several types of imperfections coexist in Knowledge-Based Systems. There is mainly uncertainty and imprecision [1]. Uncertainty means that the event realization is subject of question. However, Imprecision expresses the event vagueness to measure the degree to which an event occurs. Each imperfection type is handled by a particular theory. Indeed, uncertain knowledge are mainly processed by the probability theory [2]. Imprecision is generally treated by rough set theory [3]. Some imperfect data express both imprecision and uncertainty such as incomplete and vague information. Incompleteness means that

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 69–83, 2018.
https://doi.org/10.1007/978-3-319-98812-2_5

the probability of an event occurrence is unknown but we are sure about its possibility. They are generally treated by the possibility theory [4]. Vagueness arises from the existing of knowledge objects with concepts that have no clear boundary among others. The borderline between the definitions of these concepts is rather confused such as the information "the room is big". In this case, the concept "big" is undefined and we do not know its limits. An example of vague and uncertain information is: "Paul can be young with a confidence degree of 60%". It is precise but not a certain information. The fact that Paul is young is not at 100% guaranteed. He may be for example old. An example of vague and imprecise information is "Paul is young, adult or old". It is a certain but imprecise information. Indeed, it is certain that Paul belongs to one of the three classes {young, adult, old}, but we do not know exactly to which one. Vague knowledge are treated either by fuzzy logic [5] or by multi-valued logic [6]. Fuzzy logic is based on fuzzy set theory. Its principal is to model every term by a fuzzy set having a numerical universe. Thus, it is artificial and complicated to model abstract/qualitative data like the intelligence of the mood. This presents the main inconvenient of the fuzzy logic according to many authors [7–9]. Nonetheless, to deal with fuzzy logic, the expert has to define a membership function for each linguistic term. However, in literature, there is no standard to do it for abstract data, but each expert defines it in his own way. Modelling them with fuzzy sets is difficult and artificial because they do not refer to a numerical universe. In the other hand, symbolic multi-valued logic expresses belonging using symbolic adverbs such as *little* and *moderately*, without having to refer them to numerical universes. It is based on multi-set theory, and allows getting closer to human reasoning by a symbolic modelling.

It is likely to be in need to handle both fuzzy and multi-valued knowledge in the same Knowledge-Based System, especially when data are collected from different sources (tools or experts). Nevertheless, according to our knowledge, this type of problem has not been treated so far. The difficulty lies in the choice of the perfect unification environment that will standardize the heterogeneous inputs whether it is a numerical environment manipulated by the fuzzy logic or a symbolic environment managed by the symbolic multi-valued logic. In previous work, we initiated our research to make the right decision about the unification framework [10]. Indeed, we have proposed a *fuzzy-to-symbolic conversion* to translate fuzzy knowledge to multi-valued knowledge in Knowledge-Based Systems. We used our proposal in Rule-Based Systems, to finally infer them by employing symbolic approximate reasoning. The inference results were close to the expert reasoning. Taking account of knowledge treatment, symbolic multi-valued logic provides a simple symbolic reasoning environment. However, when a precise numerical result is required, such environment is no longer appropriate. In fact, having a multi-valued unification system can generate a loss of information when converting a fuzzy set to a multi-set. This is due to the fact that the conversion is made by a discretization of the fuzzy universe. In order to avoid this information loss and to keep mathematical characteristics of fuzzy sets, we adopt in this work the fuzzy logic as the unification environment. We propose for that an approach to automatically translate symbolic multi-valued knowledge to fuzzy knowledge: *symbolic-to-fuzzy conversion.*

The remainder of this paper is as follows: in Sect. 2, we focus on the modeling of vague data in quantitative way as well as in qualitative way by employing, respectively,

fuzzy logic and multi-valued logic. In Sect. 3, we study the generation of fuzzy sets from discrete data (numerical and symbolic) as well as from abstract data in the literature. Section 4 details our new approach; the *symbolic-to-fuzzy conversion* that consists in automatically generating a set of fuzzy sets from an initial multi-set. An example of application in Rule-Based Systems will take place in Sect. 5 to illustrate our approach. Section 6 concludes the paper and cites some perspectives.

2 Modeling and Treatment of Vague Knowledge

The vague, uncertain and imprecise knowledge can be modeled and thus processed with a numerical/quantitative way by employing the fuzzy set theory (fuzzy logic) or in a symbolic/qualitative way based on the multi-sets theory (multi-valued logic).

2.1 Modeling Vague Knowledge

The fuzzy logic that is introduced by Zadeh [5], is an extension of classical Boolean logic. In fact, it allows handling nuanced knowledge and especially those that refer to a numerical universe. A fuzzy linguistic variable is represented by the triplet <X, R (X), U> with:

- X: name of the linguistic variable.
- U: numerical universe.
- R(X): set of fuzzy sets representing X.

Fuzzy logic models quantitative knowledge through fuzzy sets that are modeled by membership functions. However, humans deal also with abstract terms that do not refer to numerical universe such as clever, beautiful, etc. In that case, he need to model such terms in a qualitative way, which is not adapted to the fuzzy set theory. As an alternative, the symbolic multi-valued logic presents another formalism to present vague data by employing the multi-set theory [6, 11]. Actually, symbolic multi-valued logic, which is an axiomatic approach of fuzzy sets theory, manipulates similarly abstract data as well as numerical data by a qualitative way. Each linguistic term is expressed by a multi-set A. The truth degree τ_α shows how much a linguistic variable X satisfies A. The membership relation between X and A is denoted by "*X is* ϑ_α *A*" where ϑ_α is an adverbial proposition associated to a symbolic degree $\tau\alpha$. As an example of a proposition, we can cite "the food is very delicious" where *food* is a linguistic variable, *very* is an adverbial proposition, and delicious is a multi-set. Employing symbolic degrees such as *quite*, *very*, and *perfectly* is more intuitive and is a part of our natural language. In fact, humans express generally their knowledge by a qualitative way rather than in a quantitative way, which represents the strength of the symbolic multi-valued logic. A degree τ_α refers to an ordered list $\mathcal{L}_M = \{\tau_\alpha, \alpha \in [0, M-1]\}$ where M is the scale size of $\mathcal{L}_M$ associated to A. For example, the multi-set delicious can be associated to $\mathcal{L}_4 = \{\tau_0, \tau_1, \tau_2, \tau_3\}$ where each multi-valued degree corresponds respectively to the following set of adverbs: {not-very, more-or-less, very, extremely}. A proportion is associated to each multi-valued degree denoted by $(\tau_i) = \frac{i}{M-1}$. As we can see, the

proportion is the rate of the degree according to the base, so $prop(\tau_i) \in [0, 1]\ \forall i$ *and* $\forall M$.

2.2 Treating Vague Knowledge

Reasoning is the process of mapping from a given inputs to an inferred output. Reasoning using fuzzy knowledge as well as symbolic multi-valued knowledge is generally ensured by the approximate reasoning [12]. Its objective is to imitate human reasoning by ensuring more flexibility in the inference step. It is used to overcome borders problem due to the discretization since it does not require a perfect conformity between facts to reason. It is based on a generalization of Classic Modus Ponens: Generalized Modus Ponens (GMP). The standard form of GMP is as follows:

Rule :	If X is A	then	Y is B
Observation:	X is A'		
Conclusion :			Y is B'

Where X, Y are linguistic variables and A, B, A', B' are predicates. GMP is adapted to be used in both fuzzy and symbolic multi-valued logics. In the former, predicates are fuzzy sets and in the latter they are multi-sets.

The commonly used GMP is proposed by Zadeh where the way in which the conclusion B' is obtained is called Compositional Rule of Inference (CRI) [13]. The form of the proposed GMP is as follows:

$$B' = A' \circ R\ (A, B) \tag{1}$$

In fact, the conclusion B' is determined as a composition of the observation and the fuzzy relation R, which is the application of a fuzzy implication operator I. The fuzzy implication I represents the fuzzy conditional statement "*X is A* $\rightarrow$ *Y is B*" and o stands for the sup-min composition of the unary relation A' and the binary relation I. The fuzzy sets A and B are defined on U and V, respectively. That is, B' is obtained by [14, 15]:

$$\forall\ v \in V, \mu_{B'}(v) = Sup_{u \in U} T(\mu_{A'}(u), \mu_I(u, v))) \tag{2}$$

Where T is a t-norm.

Fuzzy inference systems integrate fuzzy approximate reasoning in their inference engine, which contain three phases [16]. The first phase is the fuzzification that transforms a numerical input into a fuzzy point. The second phase is the inference. It refers to a knowledge base and an observation to conclude new facts and then update the knowledge base. The third phase is the Defuzzification, which is the opposite phase of fuzzification. It determines a numerical value from the fuzzy part resulting from the inference phase.

The GMP is also employed in the symbolic multi-valued context [6, 17, 18]. It is used to process a symbolic approximate reasoning that infers multi-valued knowledge. In that context, the symbolic inference system is only composed of the inference phase. Thus, comparing with fuzzy inference systems, nor the fuzzification or defuzzification

are required. Multi-valued knowledge expressed by the expert are directly integrated in the inference engine. The symbolic approximate reasoning is then employed to get a symbolic result. Different symbolic inference systems were proposed [19–22].

3 Generating Fuzzy Sets in Literature

In symbolic multi-valued logic, knowledge values correspond to a multi-valued degree τ_i. Each degree corresponds to a proportion $prop(\tau_i)$. These proportions have discrete numerical values that belong to the interval [0, 1]. Converting discrete values (numeric or symbolic) and presenting them using fuzzy sets means to accentuate their imprecision. In other words, these discrete values will be converted into fuzzy values.

In order to better understand the basis of our proposal, we outline in the following section how fuzzy sets are generated from discrete data (numeric or symbolic) and also from abstract data in the literature.

3.1 From Discrete to Fuzzy: Fuzzy Numbers

Fuzzy number was introduced by Zadeh [5]. A fuzzy number A in a universe U is defined by a membership function μ_A which specifies the degree of credibility of the assertion $x \in A$:

$$\begin{aligned} \mu : U &\rightarrow [0,1] \\ x &\rightarrow \mu_A(x) \end{aligned} \tag{3}$$

A fuzzy number A is generally defined as a trapezoidal membership function [a, b, c, d] where [a, d] is its support. This format can be used to represent a fuzzy interval (approximately between b and c) (see Fig. 1). Its function is defined as formula (4) shows [23].

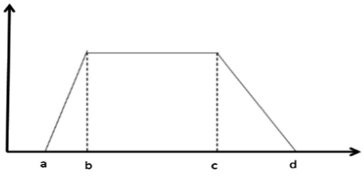


Fig. 1. Trapezoidal representation of a fuzzy number

$$(x) = \begin{cases} 1 - \frac{b-x}{b-a} \; if \; a \leq x \leq b \\ 1 \; if \; b \leq x \leq c \\ 1 - \frac{x-c}{d-c} \; if \; c \leq x \leq d \\ 0 \qquad else \end{cases} \tag{4}$$

However, some particular cases of fuzzy numbers are presented as a triangular membership function where b = c (see Fig. 2). They are commonly used to represent a value that is approximately equal to b. Its function is as formula (5) shows.

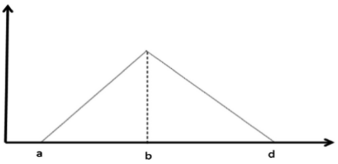


Fig. 2. Triangular representation of a fuzzy number

$$\mu_A(x) = \begin{cases} 1 - \frac{b-x}{b-a} \; if \; a \leq x \leq b \\ 1 - \frac{x-b}{d-b} \; if \; b \leq x \leq d \\ 0 \qquad else \end{cases} \tag{5}$$

3.2 From Abstract to Fuzzy

In the literature, there is no standard to model abstract data using fuzzy sets. This is actually artificial and each researcher does it in his own way. For example, in sensory analysis domain, human evaluators are used to apprehend how products are perceived. In [24], a fuzzy number, which also represents a fuzzy score, gives the evaluation of each descriptor by each expert. Every expert perception is presented by a triangular membership function to take into account the imprecision of the value in U = [0, 100]. In another work [25], the authors use to classify a car within six predefined categories. According to him, it is possible that more than one membership function form could be used to represent symbolic degrees describing the same linguistic variable. Thus, the employed estimation values are either triangular or trapezoidal fuzzy numbers within the same universe of discourse.

In [26], there is a proposal of associating to linguistic truth-values that belongs to a graduated scale L_M a membership function of a fuzzy set. Each membership function has a triangular form. The choice of such form is based on its simplicity but also in accordance with a domain expert. Authors choose a symbolic graduation from the "Not at all or small" to "Large". The universe of discourse is the interval U = [0, 255] presenting the pixel grey level. Then, membership functions are associated with a pixel feature where U is normalized to [0, 1]. Truth values are chosen in L_3 where the functions of truth degrees are respectively:

$$\mu_{Not\ at\ all\ or\ small}(x) = 1 - x$$
$$\mu_{large}(x) = x$$
$$\mu_{more\ or\ less}(x) = \begin{cases} 2x \; for \; x \leq 0.5 \\ 2 - 2x \; for \; x \geq 0.5 \end{cases}$$

However, no indication is given by the authors about the generated membership functions if more than three truth values where used. In addition, what if the linguistic variable does not belong to a numerical scale such as the grey level of an image, how should the universe of discourse of generated fuzzy sets be defined?

Similarly, in [27], there is another proposal to replace crisp membership functions by fuzzy truth-qualified statements in which the truth is a fuzzy set. Thus, authors allow expressing uncertainty such in the following statement: "Tina is young is very true". This statement may be represented with "tina:Young, ln(4)" where ln() is a linear membership function. The proposal allows representing an imprecision about the actual degree of truth. For example, it is possible to express the statement "Tina is young is

true to degree around 0.7", as an axioms of the form "tina:Young, tri(0.6, 0.7, 0.8)", where tri() is a triangular membership function.

Consequently, there is no standard to generate automatically, without the recommendation of an expert, fuzzy sets from abstract data. The difficulties lie first in the choice of the interval of discourse where membership functions of fuzzy sets will be defined, and second in the form of each fuzzy set.

4 The Proposed Approach: Unifying Fuzzy and Multi-valued Knowledge

The problematic that we are dealing with is how to integrate multi-valued and fuzzy knowledge in the same Knowledge-Based System. We adopt as an example of application Rule-Based Systems. In these systems, premises, rule conclusions, and even observations may contain fuzzy and multi-valued predicates at the same time. Fuzzy predicates are modeled by fuzzy sets whereas multi-valued predicates are modeled by multi-sets. We cite as an example the following rule: "if the student is intelligent then its exam mark is high" where *student* and *exam mark* are two linguistic variables, *intelligent* is a multi-set, and *high* is a fuzzy set.

Inferring such heterogeneous knowledge requires first to integrate them into a unified environment: either fuzzy or multi-valued. As explained above, the multi-valued logic is perfect for modeling quantitative and qualitative knowledge. However, it generates a loss of information with quantitative knowledge due to the discretization [10]. Nonetheless, the fuzzy logic that is characterized by a rigorous mathematical environment presents a perfect tool to handle imprecise knowledge. For these reasons, we propose to model multi-valued knowledge by employing the fuzzy set theory. The generation of fuzzy sets procedure that we propose is advantageous compared to the other works [24–26] by being automatic and it does not require the intervention of an expert. This is done by relying on the characteristics of the symbolic degrees in the multi-valued logic, more precisely their proportions. Consequently, the standardized knowledge will be inferred by adopting the fuzzy approximate reasoning in order to take advantage of the accuracy of the results.

In order to standardize heterogeneous inputs, we propose to convert multi-valued data into fuzzy data by applying the *symbolic-to-fuzzy conversion*. In other words, we propose to translate each multi-set to a set of fuzzy sets. We also apply the *symbolic-to-fuzzy conversion* if the object that we want to evaluate has a multi-valued type, that is, if the rule conclusion contains a multi-valued predicate. Once standardized, fuzzy approximate reasoning can be performed leading to fuzzy inference conclusions. These conclusions are then aggregated to get finally a new resulting membership function. The defuzzification will then take place to discretize it into a single value. If the predicate of the rule conclusion is originally fuzzy, then there is nothing to do and the result is delivered to the user as it is. In the case where the predicate is originally multi-valued, then we propose to apply the *fuzzy-to-symbolic conversion*. This method will convert the numerical fuzzy output into symbolic multi-valued value. The aim of the *fuzzy-to-symbolic conversion* is to make the inference conclusion intelligible to the user and to guarantee that the *symbolic-to-fuzzy conversion* will be completely transparent

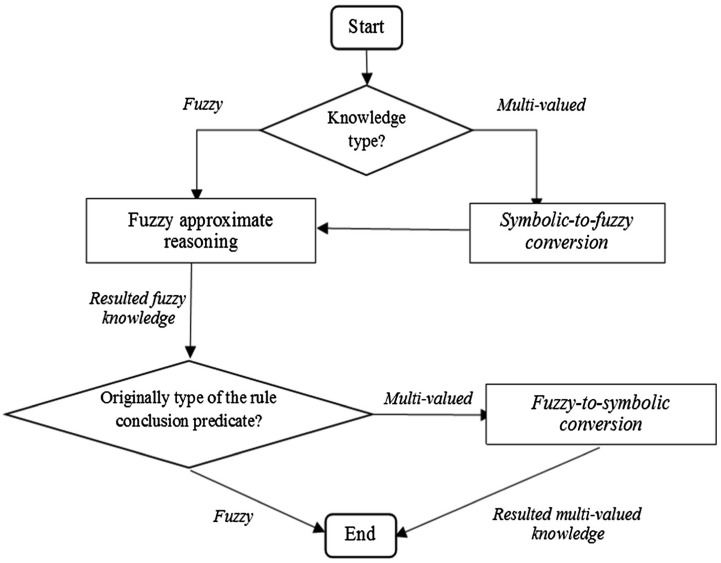


Fig. 3. A general algorithm of fuzzy approximate reasoning with fuzzy and multi-valued knowledge

for him. Figure 3 schematizes different steps, which are detailed in the sections below. In this paper, we apply the *symbolic-to-fuzzy conversion* module in Rule-Based Systems. However, it can be easily used in any Knowledge-Based System.

4.1 Symbolic-to-Fuzzy Conversion

In order to convert fuzzy knowledge into multi-valued knowledge, two steps are required: (a) Generation of the universe of discourse, (b) Generation of fuzzy sets.

Generation of the Universe of Discourse. A linguistic variable is defined over a multi-valued base where its value is a multi-valued degree such as little, very, and completely. For each degree is associated a proportion $prop(\tau_i) = \frac{i}{M-1}$. Consequently, the minimum value that can take $prop(\tau_i)$ is when i = 0 then $prop(\tau_0) = 0$. The maximum value is attain when i = M−1, then $prop(\tau_{M-1}) = 1$. Consequently, the proportions over the multi-valued base are distributed over a scale having as interval [0;1]. In our approach, we propose that this scale presents the abscissa axis of the generated membership functions. In other words, the generated universe of discourse of fuzzy sets is the interval U = [0;1].

Generation of Fuzzy Sets. Once the universe of discourse is ready, we have to generate the fuzzy sets. We propose to generate from each multi-valued degree in $\mathcal{L}_M$ a fuzzy set that is represented by a membership function F_i. Consequently, we get as many fuzzy sets as multi-valued degrees in the initial multi-valued base $\mathcal{L}_M$:

$$Multi - set_{\mathcal{L}_M} \mapsto Fuzzy\ sets\{F_0, \ldots, F_{M-1}\} \tag{6}$$

To do this, we consider the proportion value of each multi-valued degree as a fuzzy number. In order to extend its range of inaccuracy and to simplify its mathematical manipulation, we transform each proportion into a triangular membership function (see Fig. 4). The tracing of each membership function is based on three points P_1^i, P_2^i (the central point of the triangular function), and P_3^i whose definitions are as follows:

$$P_1^i = \left(\max\left(0, \frac{i-2}{M-1}\right), 0\right); P_2^i = \left(\frac{i}{M-1}, 1\right); P_3^i = \left(\min\left(1, \frac{i+2}{M-1}\right), 0\right) \tag{7}$$

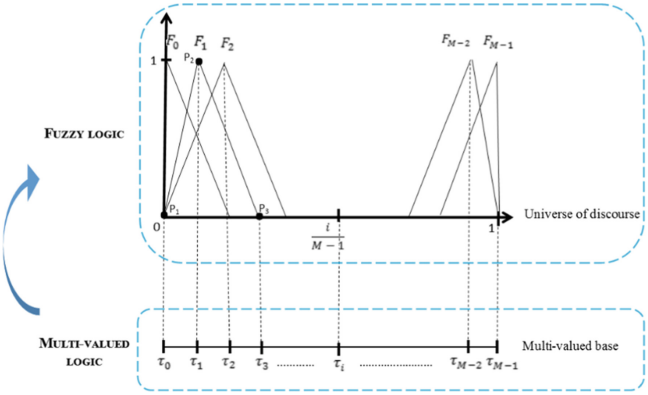


Fig. 4. Translation from multi-valued logic to fuzzy logic

We aim through the proposed triangular form to extend the imprecision of multi-valued degrees and to ensure a maximum of overlap area between the generated fuzzy sets. Thus, the generated universe of discourse will be better covered to enhance decision-making process. For example, considering a multi-set A defined by the multi-valued base $\mathcal{L}_7$. Then, the generated fuzzy sets are as follows:

$$A_{\mathcal{L}_7} \mapsto Fuzzy\ sets\{F_0, F_1, F_2, F_3, F_4, F_5, F_6\}$$

Where the tracing of each fuzzy set rely on the following three points:

$F_0\left(P_1^0(0,0), P_2^0(0,1), P_3^0(0.33,0)\right)$;
$F_1\left(P_1^1(0,0), P_2^1(0.16,1), P_3^1(0.5,0)\right)$;
$F_2\left(P_1^2(0,0), P_2^2(0.33,1), P_3^2(0.66,0)\right)$;
$F_3\left(P_1^3(0.16,0), P_2^3(0.5,1), P_3^3(0.83,0)\right)$;
$F_4\left(P_1^4(0.33,0), P_2^4(0.66,1), P_3^4(1,0)\right)$;
$F_5\left(P_1^5(0.5,0), P_2^5(0.83,1), P_3^5(1,0)\right)$;
$F_6\left(P_1^6(0.66,0), P_2^6(1,1), P_3^6(1,0)\right)$.

4.2 Fuzzy Approximate Reasoning

Once heterogeneous inputs are standardized into fuzzy type, the fuzzy approximate reasoning can be performed to infer new knowledge. We choose to use the most basic scheme of the generalized modus ponens, which is the Compositional Rule of Inference (CRI) of formula (2) proposed by Zadeh [13].

In order to simplify the inference process, we consider in this paper only numerical inputs such as in fuzzy controller. Indeed, if the input is originally fuzzy, then a numerical value u ∈ U will be considered and then fuzzified. Otherwise, if the input knowledge is originally multi-valued, then the value of its proportion will be considered and then fuzzified. For example, let $\mathcal{L}_3 = \{\tau_0; \tau_1; \tau_2\}$, if the observation is «X is τ_1 A» then the input that will be taken into consideration is the proportion value of the degree τ_1, which is equal in this case to prop(τ_1) = 1/(3−1) = 0.5. This value will be then fuzzified and will trigger CRI (2).

After inference phase, we get a resulting fuzzy part that aggregates the conclusions of the triggered rules. This fuzzy part is then defuzzified. We choose to perform the defuzzification step by relying on the centroid method [28]. If the rule conclusion is fuzzy then the result of the defuzzification is delivered to the user as it is. Otherwise, if the rule conclusion has a multi-valued type, then we propose a *fuzzy-to-symbolic* conversion to convert the numerical value resulting from the defuzzification to an equivalent symbolic value.

4.3 Fuzzy-to-Symbolic Conversion

The interest of the *fuzzy-to-symbolic conversion* is to make intelligible to the user the inference result. In addition, in a fuzzy Rule-Based System, it allows the transparency of the conversion of multi-valued knowledge. The *fuzzy-to-symbolic conversion* use to find the nearest proportion to the defuzzified value u_G. Once found, it returns to the user its corresponding multi-valued degree τ_i. To do this, we need to calculate then compare all the distances between u_G and $prop(\tau_i)$ to find the minimum distance d_{min}:

$$\forall\, i \in [0, M-1], d_{min} = \min_i(|u_G - prop(\tau_i)|) \tag{8}$$

Let consider, as an example, the fuzzy partition presented in Fig. 5, which is generated from the multi-valued base $\mathcal{L}_7 = \{\tau_0; \tau_1; \tau_2; \tau_3; \tau_4; \tau_5; \tau_6\}$. Suppose that the defuzzification of the inference result using the center of gravity method produces the value $u_G = 0.4$. To determine which proportion is the closest from u_G, we proceed as follows:

$$\begin{aligned}
&\forall i \in [0, 6], d_{min} = \min(|0.4 - 0|, |0.4 - 0.16|,\\
&|0.4 - 0.33|, |0.4 - 0.5|,\\
&|0.4 - 0.66|, |0.4 - 0.83|, |0.4 - 1|\\
&= \min(0.4, 0.24, 0.07, 0.1, 0.26, 0.43, 0.6) = 0.07
\end{aligned}$$

Consequently, the smallest distance is between u_G and $prop(\tau_2)$. So the final result is τ_2.

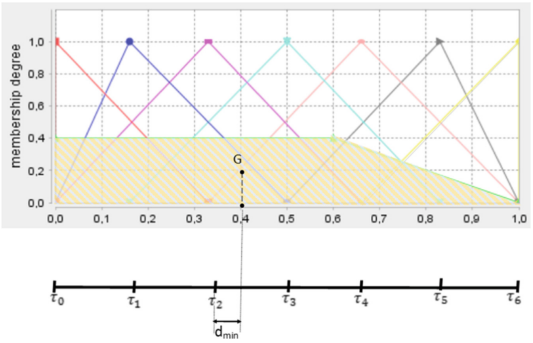


Fig. 5. The Fuzzy-to-symbolic conversion of fuzzy inference result u_G

5 Example of Application

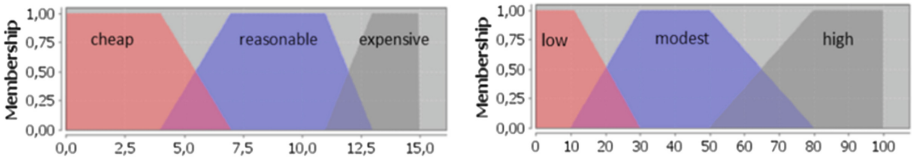
To explain how our approach is operating, we propose a complete example of an heterogeneous Rule-Based System. We consider the following rule base of a store where a detergent is sold out.

RULE 1: IF product is very efficient AND price is reasonable THEN sales are high;
RULE 2: IF product is little-bit efficient THEN sales are modest;
RULE 3: IF product is little-bit efficient OR price is expensive THEN sales are low;
RULE 4: IF sales are high THEN the marketing is extremely successful;
RULE 5: IF sales are modest THEN the marketing is little-bit successful;

- product and marketing are linguistic variables evaluated respectively by the multi-sets efficient and successful. Both multi-sets are defined by the multi-valued base $\mathcal{L}_4$ = {little-bit; more-or-less; very; extremely}.
- price (€) and sales (M €) are linguistic variables evaluated by fuzzy sets. Their graphical partitions are depicted, respectively, in Fig. 6a and b.

Let consider the following observations:

- OBSERVATION 1: Product is more-or-less efficient.
- OBSERVATION 2: Price is equal to 15€.



a. Graphical partition of price (€) b. Graphical partition of sales (M€)

Fig. 6. Graphical partition of fuzzy inputs

5.1 Symbolic-to-Fuzzy Conversion

Since facts in the rule base contain both fuzzy and multi-valued knowledge, we should apply the *symbolic-to-fuzzy conversion* to translate the multi-sets *successful* and *efficient* to fuzzy sets.

Generation of the Universe of Discourse. The generated universe of discourse of the new fuzzy sets will be the interval U = [0, 1].

Generation of Fuzzy Sets. Since both multi-sets (*successful* and *efficient*) are defined over the multi-valued base $\mathcal{L}_4$, then from each multi-set we generate four fuzzy sets (formula (6)). Then:

$$\begin{aligned} Efficient_{\mathcal{L}_4} \mapsto \{F_0 = Little - bit; F_1 = more - or - less; \\ F_2 = very; F_3 = extremely\} \\ Successful_{\mathcal{L}_4} \mapsto \{F_0 = Little - bit; F_1 = more - or - less; \\ F_2 = very; F_3 = extremely\} \end{aligned}$$

The tracing of both sets of fuzzy sets is performed by the same manner since both are based on the same multi-valued base $\mathcal{L}_4$ (see Fig. 7). For example, the tracing of F_0 according to formula (7) is as follows:

$$\begin{aligned} P_1^0 &= \left(\max\left(0, \frac{i-2}{M-1}\right), 0\right) = \left(\max\left(0, \frac{0-2}{4-1}\right), 0\right) = (0,0) \\ P_2^0 &= \left(\frac{i}{M-1}, 1\right) = (0,1) \\ P_3^0 &= \left(\min\left(1, \frac{i+2}{M-1}\right), 0\right) = \left(\min\left(1, \frac{0+2}{4-1}\right), 0\right) = \left(\frac{2}{3}, 0\right) \end{aligned}$$

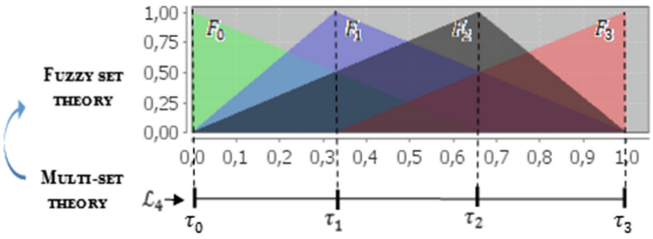


Fig. 7. Generation of fuzzy sets from $\mathcal{L}_4$

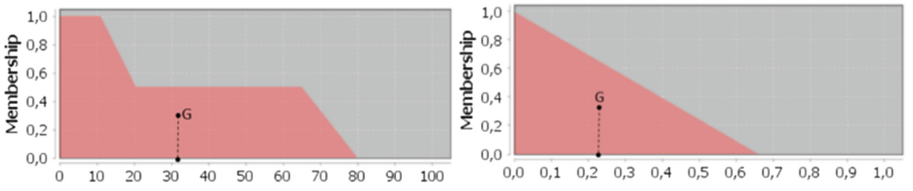
5.2 Fuzzy Approximate Reasoning

Once heterogeneous knowledge are unified, we can execute the fuzzy approximate reasoning. To infer, we rely on an open source Java library called jFuzzyLogic [29], which offers a fully functional and complete implementation of a fuzzy inference

system. We employ the MIN and MAX aggregation methods respectively for the operators AND and OR. We also use MIN as an implication operator and to aggregate all the rule conclusion, we use MAX method.

Using fuzzy reasoning, we get as an inference result of rules 1, 2 and 3 with the observation 1 and 2: "u_{sales} = 31.3 M€" (see Fig. 8a). The result is fuzzy, as the original type of the linguistic variable *sales*. Therefore, no conversion is necessary.

When inferring rules 3 and 4 with the previous inference result as observation, we get "$u_{marketing}$ = 0.22" (see Fig. 8b). This result is not intelligible for the user since values that he knows about how much the marketing is successful, are only symbolic and not numeric. Indeed, the original type of the linguistic variable *marketing* is multi-valued. For that reason, it is required to apply the *fuzzy-to-symbolic conversion*.



a. Inference result of OBSERVATION 1 and 2

b. Inference result of the OBSERVATION "usales =31.3 M€"

Fig. 8. Inference results

5.3 Fuzzy-to-Symbolic Conversion

We need at this phase to find the nearest proportion value belonging to $\mathcal{L}_4$ to the defuzzified value $u_{marketing}$ = 0.22. For that purpose, we rely on formula (8), so we get:

$$\forall\, i \in [0,3], d_{min} = \min_{i} \left(\left|u_{marketing} - prop(\tau_i)\right|\right) = 0.11$$

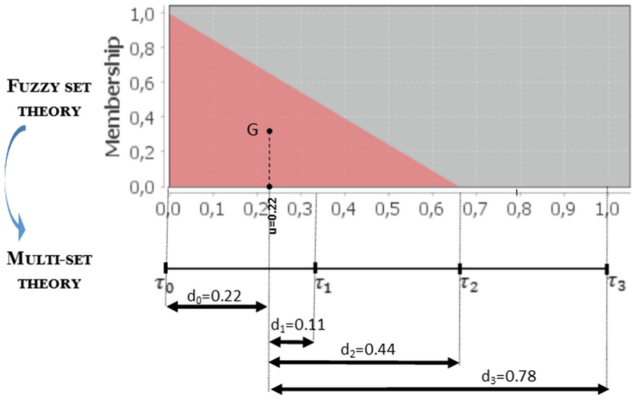


Fig. 9. Fuzzy-to-symbolic conversion of $u_{marketing}$ = 0.22

Consequently, as shown in Fig. 9, the nearest proportion from $u_{marketing}$ is $prop(\tau_1)$. The degree τ_1 corresponds in $\mathcal{L}_4$ to the linguistic term "more-or-less". Then the inference result returned to the user is «The marketing is more-or-less successful».

6 Conclusion

In this paper, we propose a new modeling and standardization approach of fuzzy and symbolic multi-valued knowledge. As a unified framework, we adopt the fuzzy logic, which provide a rigorous mathematical theory to handle imprecise knowledge. In that context, we propose a *fuzzy-to-symbolic conversion* method to translate each initial multi-set to a set of generated fuzzy sets. We apply our approach in Rule-Based System, which offers a high flexibility to the user to exploit heterogeneous knowledge simultaneously regardless to the manner that they were evaluated, if it was using fuzzy set theory or in multi-set theory. In addition, we propose a *symbolic-to-fuzzy conversion* method that is able to reset inference result to its initial type if it was multi-valued. Consequently, the *fuzzy-to-symbolic conversion* will be completely transparent for the user. In future work, we are going to evaluate the performance of our proposal with a real rule-base and compare results with our previous work [10] where the unified framework was the symbolic multi-valued logic. The aim of that comparison is to decide of the optimal unification environment to handle both fuzzy and multi-valued knowledge.

References

1. Smets, P.: Imperfect information: imprecision and uncertainty. In: Motro, A., Smets, P. (eds.) Uncertainty Management in Information Systems, pp. 225–254. Springer, Boston (1997). https://doi.org/10.1007/978-1-4615-6245-0_8
2. Jaynes, E., Bretthorst, G.: Probability Theory: The Logic of Science. Cambridge University Press, Cambridge (2013)
3. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Norwell (1992)
4. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst. **100**, 9–34 (1999)
5. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**, 338–353 (1965)
6. Akdag, H., De Glas, M., Pacholczyk, D.: A qualitative theory of uncertainty. Fundam. Inform. **17**, 333–362 (1992)
7. De Glas, M.: Knowledge representation in a fuzzy setting. Internal report 48, LAFORIA, University of Paris VI (1989)
8. Pacholczyk, D.: Contribution au traitement logico-symbolique de la connaissance. Ph.D. thesis, University of Paris VI (1992)
9. Chung, H., Schwartz, D.: A resolution-based system for symbolic approximate reasoning. Int. J. Approx. Reason. **3**, 201–246 (1995)
10. Moussa, S., Kacem, S.B.H.: Symbolic approximate reasoning with fuzzy and multi-valued knowledge. In: International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES, Marseille, France, vol. 112, pp. 800–810 (2017)

11. Ginsberg, M.: Multivalued logics: a uniform approach to reasoning in artificial intelligence. Comput. Intell. **3**, 265–316 (1988)
12. Zadeh, L.A.: A theory of approximate reasoning. Mach. Intell. **9**, 149–194 (1979)
13. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning-I. Inf. Sci. **8**, 199–249 (1975)
14. Didier, D., Henri, P., Laurent, U.: Base de règles floues en commande: une discussion critique. Technical report, IRIT, Toulouse, 59-48-r (1995)
15. Dubois, D., Prade, H.: What are fuzzy rules and how to use them. Fuzzy Sets Syst. **84**, 169–185 (1996)
16. Klir, G.: A review of: adaptive fuzzy systems and control: design and stability analysis, by Li-Xin Wang. In: Prentice Hall, Englewood Cliffs, N.J., 1994. XVII+ 232 p. Int. J. Gen. Syst. **25**, 177–178 (1996)
17. Khoukhi, F.: Approche logico-symbolique dans le traitement des connaissances incertaines et imprécises dans les systèmes à base de connaissances. Doctoral thesis. University of Reims, France (1996)
18. Kacem, S.B.H., Borgi, A., Tagina, M.: Extended symbolic approximate reasoning based on linguistic modifiers. Knowl. Inf. Syst. **42**, 633–661 (2014)
19. Phuong, L., Khang, T.: Generalized modus tollens with linguistic modifiers for inverse approximate reasoning. Int. J. Comput. Intell. Syst. **7**, 556–564 (2013)
20. Phuong, L., Khang, T.D.: Linguistic reasoning based on generalized modus ponens with linguistic modifiers and hedge moving rules. In: International conference on Fuzzy Theory and Its Applications, iFUZZY2012, pp. 82–86 (2012)
21. Kacem, S.B.H., Borgi, A., Othman, S.: A diagnosis aid system of autism in a multi-valued framework. In: Uncertainty Modelling in Knowledge Engineering and Decision Making (FLINS), pp. 405–410 (2016)
22. Chaoued, N., Borgi, A., Laurent, A.: Camphor odor recognition within unbalanced multi-sets. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–6 (2017)
23. de Barros, L.C., Bassanezi, R.C., Lodwick, W.A.: The extension principle of zadeh and fuzzy numbers. In: de Barros, L.C., Bassanezi, R.C., Lodwick, W.A. (eds.) A First Course in Fuzzy Logic, Fuzzy Dynamical Systems, and Biomathematics. SFSC, vol. 347, pp. 23–41. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-53324-6_2
24. Hébert, P.A., Masson, M.H., Denux, T.: Fuzzy multidimensional scaling. Comput. Stat. Data Anal. **51**, 335–359 (2006)
25. Yang, J.B., Wang, Y.M., Xu, D.L., Chin, K.S.: The evidential reasoning approach for MADA under both probabilistic and fuzzy uncertainties. Eur. J. Oper. Res. **171**, 309–343 (2006)
26. Borgi, A., Akdag, H.: Knowledge based supervised fuzzy-classification: an application to image processing. Ann. Math. Artif. Intell. **32**(1/4), 67–86 (2001)
27. Fernando, B., Umberto, S.: Fuzzy descriptions logics with fuzzy truth values. In: Proceedings of the World Congress of the International Fuzzy Systems Association/European society for Fuzzy Logic and Technology (IFSA/EUSFLAT 2009), pp. 189–194 (2009)
28. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **SMC-15**, 116–132 (1985)
29. Cingolani, P., Alcalá-Fdez, J.: jFuzzyLogic: a Java library to design fuzzy logic controllers according to the standard for fuzzy control programming. Int. J. Comput. Intell. Syst. **6**, 61–75 (2013)

Frequent Itemset Mining on Correlated Probabilistic Databases

Yasemin Asan Kalaz(✉) and Rajeev Raman

University of Leicester, Leicester LE1 7RH, UK
{ya63,r.raman}@leicester.ac.uk

Abstract. The problem of mining frequent itemsets from uncertain data (uFIM) has attracted attention in recent years. Most of the work in this field is based on the assumption of stochastic independence, which is clearly unjustified in many real-world applications of uFIM. To address this problem, we introduce a new general model for expressing dependencies in frequent itemset mining. We show that mining itemsets in the general model is NP-complete, but give an efficient algorithm based on dynamic programming to mine itemsets in a simplified version of this model. Our experimental results show that assuming independence in correlated data sets leads to substantially incorrect results.

1 Introduction

Frequent itemset mining (FIM) is a fundamental problem in data mining. However, it has been argued that a large amount of data generated by emerging technologies such as RFID and networks [12], information extraction services [5], etc. is *uncertain*. In these technologies, data uncertainty occurs due to sensor and network errors and aggregation of incomplete or inconsistent data. Another source of "uncertainty" is as a summary of a large volume of certain data; examples of this include PWM (position-specific weight matrices) in bioinformatics [7] and shopper profiles [3]. Such uncertainties are often modeled using the *probabilistic database* framework [9]. The FIM problem formulated on probabilistic databases is known as uncertain frequent itemset mining (uFIM) and has attracted much attention in recent years [3,4,10,11,13]. In classical FIM, we want to mine a *transaction database*, which contains a list of *transactions*, each of which contains a set of items (in a retail scenario, an e.g. of this could be items purchased by a shopper). In uFIM, we are given a *probabilistic transaction database* (PDB), where each item in a transaction is labeled with a probability, which is interpreted as the probability of that item existing in the transaction (an interpretation of this in the retail scenario could be the probability of a customer buying a given item). An example PDB is shown in Fig. 1.

PDBs are interpreted using *possible world semantics* [9]. The most common way of generating the possible worlds is by assuming *independence* among the probabilistic items in each transaction in a PDB [3,10,13]. In the scenario used by Bernecker et al., the independence assumption applied to the example of Fig. 1a

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 84–98, 2018.
https://doi.org/10.1007/978-3-319-98812-2_6

ID	Transaction
t_A	{Game(1.0), Music(0.2)}
t_B	{Video(0.4), Music(0.7)}

(a)

	Possible World	Prob
W_1	$\{t_A.Game\}$	0.144
W_2	$\{t_A.Game, t_A.Music\}$	0.036
$\cdots$	$\cdots$	$\cdots$
W_8	$\{t_A.Game, t_A.Music, t_B.Music, t_B.Video\}$	0.056

(b)

Fig. 1. (a) shows a sample PDB from [3], where the probability of each item is the likelihood that the customer will buy the item on his/her store visits. This PDB has eight possible worlds. (b) shows a subset of possible worlds for the PDB of (a), calculating probabilities under the independence assumption.

would mean that customers A and B purchase items independently of each other: buying decisions made by A are not influenced by the buying decisions of B and vice versa. However, it may be that in real life A and B know each other, and the decision of (say) A to buy (or not) music influences B's decision. If this is the case then the probabilities assigned to the possible worlds by the independence assumption may be incorrect.

The uFIM problem comes in two variants: *expected support* [4] and *probabilistic frequentness* [3]. The latter is normally preferred because it provides much more precise (and actionable) information. Unfortunately, the probabilities assigned to individual possible worlds are critical for computing probabilistically frequent itemsets, and answers computed according to the independence assumption can lead to incorrect results. We further explain this using Example 1 in Sect. 2.

In this paper, we define a general model, the *directed acyclic graph correlated probabilistic database model* (DAG CPDM) for uFIM that allows dependencies to be captured. This model generalizes the existing independence model, and allows the existence of an item in a transaction to be decided by a probabilistic DAG. It turns out that this model is *too* general: we show that deciding whether an individual itemset is probabilistically frequent is NP-complete. On the positive side, we propose a restricted version of the above model, the *directed acyclic graph-restricted correlated probabilistic database model* (DAG-R CPDM) and give a dynamic-programming algorithm for support computation for this model. Embedding this into the Apriori algorithm, we can efficiently enumerate all probabilistically frequent itemsets when correlations are expressed in the DAG-R CPDM model. Our experiments show that on PDBs where there are correlations between items, there are significant differences between the itemsets computed by our algorithm and those obtained by ignoring these correlations and assuming independence.

Overview. The rest of this paper is organized as follows. We present problem formulation in Sect. 2. Sections 3 and 4 introduce the DAG CPDM and DAG-R CPDM respectively. Related work is presented in Sect. 5. Finally, we present experimental evaluation in Sect. 6 and conclude in Sect. 7.

2 Problem Formulation

In this section, we first present formal definition of probabilistic transaction databases, then we define possible world semantics. Next, we define frequentness probability and probabilistic frequent itemset.

Probabilistic Transaction Databases (PDB). A probabilistic transaction database R is a set of transactions $\{t_1, t_2, \ldots, t_n\}$, where each transaction is denoted by a tuple. A tuple consists of a unique tuple id tID, and an itemset Y, i.e. $t_i = (tID, Y)$, where $Y = \{y_1(p_1), y_2(p_2), \ldots, y_r(p_r)\}$. Each element of the set Y has an item y_s and a probability of that item p_s, and p_s is the probability that y_s "truly" exists in t_i. An example of PDB can be found in Fig. 1a.

Possible World Semantics. A PDB is considered as a set of deterministic database instances, each of which is called a possible world. Each possible world W has zero or more tuples, and has a probability value associated with it. The probability value of a possible world W is denoted by $\Pr(W)$, it shows how likely W is the true world. The sum of the probability values of all possible worlds equals 1. Under the independence assumption, each item in each transaction is considered to be present or absent with the appropriate probability, independently of all other items in the PDB. The possible worlds of the PDB of Fig. 1a are given in Fig. 1b, with probabilities calculated according to the independence assumption. For example the possible world $\{t_A.Game\}$ means that only the item $Game$ in t_A is present, and all other items are not present. Its probability value is computed as follows. $1.0 \times (1 - 0.2) \times (1 - 0.4) \times (1 - 0.7) = 0.144$.

Definition 1 ***(Frequentness Probability** [3,13]**).*** *Given a PDB R, a minimum support threshold θ, the frequentness probability of an itemset X, denoted as $P_F(X)$, is defined as:*

$$P_F(X) = \Pr[Support(X) \geq \theta]$$

Definition 2 ***(Probabilistic Frequent Itemset** [3,13]**).*** *Given a PDB R, a minimum support threshold θ, and a probabilistic threshold δ, an itemset X is a probabilistic frequent itemset if $P_F(X) \geq \delta$.*

Example 1. We now argue that when computing probabilistic frequentness, it is essential to model dependencies correctly. Looking now at the PDB in Table 1, we focus only on the item "music", whose support could be $0, 1$, or 2, and compute the probability distribution of the support of music under three assumptions, ignoring the other items[1]. We abbreviate $t_A.Music$ and $t_B.Music$ as $A.M$ and $B.M$ below. In each case below, we only fully compute $\Pr[Support(\{Music\} = 1)]$; the full results are in Table 1.

1. (Independent) Assuming $A.M$ and $B.M$ are independent, $\Pr[Support(\{Music\}) = 1] = 0.7 \times (1 - 0.2) + 0.2 \times (1 - 0.7) = 0.62$.

[1] Mathematically, we assume that $t_A.Music$ and $t_B.Music$ are independent of $t_A.Game$ and $t_B.Video$.

2. (Positive Correlation) Customer A only buys music if Customer B buys it. Specifically, $\Pr[B.M] = 0.7$, but $\Pr[A.M|B.M] = 2/7$, and $\Pr[A.M|\neg B.M] = 0$. In this case $\Pr[A.M] = \Pr[A.M \wedge B.M] = \Pr[A.M|B.M]\Pr[B.M] = 0.2$ as before, but $\Pr[Support(\{Music\}) = 1] = \Pr[B.M \wedge \neg A.M) = 0.7 \times (1 - 2/7) = 0.5$.
3. (Negative Correlation) Customer A only buys music if Customer B does *not* buy it. Specifically, $\Pr[B.M] = 0.7$, but $\Pr[A.M|B.M] = 0$, and $\Pr[A.M|\neg B.M] = 2/3$. In this case $\Pr[A.M] = \Pr[A.M \wedge \neg B.M] = \Pr(A.M|\neg B.M)\Pr(\neg B.M) = 0.2$ as before, but $\Pr[Support(\{Music\}) = 1] = \Pr(A.M \wedge \neg B.M) + \Pr[\neg A.M \wedge B_M] = 0.7 + 0.2 = 0.9$.

Table 1. The distribution of the support of $\{Music\}$ from the PDB of Fig. 1a.

Model	Support		
	0	1	2
Independent	0.24	0.62	0.14
DAG-R positive correlation	0.3	0.5	0.2
DAG-R negative correlation	0.1	0.9	0

Now assume that $\theta = 1$. According to the probability distribution that we have in Table 1, $P_F(Music)$ is $0.62 + 0.14 = 0.76$, $0.5 + 0.2 = 0.7$ and 0.9 respectively, depending on whether the possible worlds are calculated according to independence, positive correlation, or negative correlation. Thus, for example, using $\delta = 0.75$, $\{Music\}$ is probabilistic frequent assuming independence but not frequent assuming positive correlation. Using $\delta = 0.8$, $\{Music\}$ is not probabilistic frequent assuming independence, but is frequent assuming negative correlation. This shows that by using independence assumption we may not find the correct probabilistic frequent itemsets.

3 DAG-Correlated Probabilistic Database Model

3.1 Overview

In this model, there are correlations between individual items in transactions. These correlations are limited to *acyclic* dependencies. The model is defined as follows. We are given a PDB $R = \{t_1, t_2, \ldots, t_n\}$, where each t_i is a tuple. We will use the notation $t_i.Y$ to denote the itemset of tuple t_i, and $t_i.y_j$ to denote an individual item in tuple t_i, which we later call a *component*. In addition, we have a set of predefined *correlation rules*, which is defined as $G = \{C_1, C_2, \ldots, C_m\}$. These correlation rules are defined over a dependency graph $G_d = (V, E)$ where the set of vertices V is $\cup_{i=1}^{n} \cup_{j=1}^{|t_i.Y|} t_i.y_j$. In other words, each possible component of a tuple is a vertex of this graph. The dependency graph is assumed to be

acyclic and *bounded in-degree*. In what follows, we will interchangeably refer to a vertex v and the component of a tuple that it represents.

The correlation rules are specified as follows. Consider any vertex $v \in V$ and suppose that edges $(u_1, v), \ldots, (u_k, v)$ exist. Then, in this correlation rule, we are given 2^k conditional probabilities for the existence of v, given the existence or non-existence of each of $u_1, \ldots, u_k$. If there are no edges of the form (u, v) then v is assumed to be independent of all other vertices in G_d, and there is no correlation rule for v.

Possible World Semantics. Given a PDB R, and a set of correlation rules G, we define a possible world as a set of boolean variables, where each boolean variable corresponds to a vertex. If the boolean variable has true value then the corresponding vertex exists, otherwise the vertex does not exist. A possible world denoted as $W = \{\delta_1, \delta_2, \ldots, \delta_n\}$. The probability values of the possible worlds can be computed by the following formula.

$$\begin{aligned} \Pr(W) &:= \Pr(v_1 = \delta_1, v_2 = \delta_2, \ldots, v_{|V|} = \delta_{|V|}) \\ &= \Pi_{i=1}^{|V|} \Pr(v_i = \delta_i | \{v_j = \delta_j \text{ such that } (v_i, v_j) \in E\}) \end{aligned} \tag{1}$$

where $\Pr(v_i = \delta_i | \{v_j = \delta_j\})$ shows conditional probability between vertex v_i and v_j. To compute all the conditional probabilities we apply chain rule.

3.2 Computational Complexity

Probabilistic Support Problem. Let D be a DAG correlated PDB, n be the number of transactions, G be a set of m correlation rules, θ be the support threshold, where $0 \leq \theta \leq n$ and δ be a probabilistic threshold . The probabilistic support problem is, given D, s, delta, and an itemset X, to decide whether $P[Support(X, D) \geq \theta] \geq \delta$, i.e. the probability value that at least θ number of transactions support X with at least δ probability value, or not in polynomial time.

Theorem 1. *The probabilistic support problem is NP-hard.*

Proof. We reduce *3-SAT* problem to the probabilistic support problem. Let $F = (x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge \ldots \wedge (x_{m,1} \vee x_{m,2} \vee x_{m,3})$ be a *boolean formula* with m clauses in conjunctive normal form (CNF) over boolean variables $y_1, \ldots, y_n$, where each $x_{i,j}$ is a literal that equals some variable y_k or its negation $\overline{y_k}$ (we assume that $x_{i,1}, x_{i,2}$, and $x_{i,3}$ all refer to distinct variables). The *3-SAT* problem asks whether, given such an F, there is an assignment of values to $y_1, \ldots, y_n$ such that F evaluates to true.

Given a *3-SAT* formula F, we create a DAG correlated PDB D in polynomial time such that solving exact support problem gives solution to the *3-SAT* problem. For the given formula F, we create $2n + m$ tuples with n t_i and n f_i tuples. The tuples t_i and f_i correspond to y_i and its negation $\overline{y_i}$ respectively.

The tuples t_i and f_i both contain just $\{a(0.5)\}$ for some item a. In addition, we generate tuples $e_1, \ldots, e_m$, where e_i corresponds to the ith clause in F. The tuples e_i all contain $\{a(0.875)\}$, for the same item a.

Next, we create correlation rules $C_1, \ldots, C_n$, where C_i links $t_i.a$ to $f_i.a$, and specifies that their joint probability is 0. Thus, every possible world must have exactly one of $t_i.a$ or $f_i.a$. In addition, we create correlation rules $C'_i, \ldots, C'_m$ where C'_i corresponds to the ith clause. For the vertex $e_i.a$, if the j-th literal $x_{i,j}$ equals y_k (resp. $\overline{y_k}$), then there is an edge from $t_k.a$ (resp. $f_k.a$) to $e_i.a$ Thus, $e_i.a$ has a total of three predecessors in the DAG. We set the conditional probabilities for $e_i.a$ in the natural way, best illustrated by an example. If $e_i = y_3 \vee \overline{y_5} \vee y_7$, then the probability of $e_i.a$ existing, conditioned on the existence of any of $t_3.a$, $f_5.a$, or $t_7.a$, is 1, and conditioned on the existence of *none* of $t_3.a$, $f_5.a$, or $t_7.a$, is 0. Since the probability of each of the predecessors of $e_i.a$ is 0.5 and they are independent of each other, the probability of $e_i.a$ is $1 - (0.5)^3 = 0.875$.

Since the existence of $e_i.a$ is fully determined by the existence of the $t.a$ and $f.a$ tuples, there are in fact only 2^n possible worlds for D. Each possible world has probability 2^{-n}, corresponds to a truth assignment, and in each possible world, either $t_i.a$ or $f_i.a$ exists. For every satisfying truth assignment, all of the $e_i.a$'s exist, and for any non-satisfying truth assignment, at least one of the $e_i.a$'s does not exist. Thus, in all possible worlds corresponding to satisfying assignments, the support of the item a is $n + m$, and in all possible worlds corresponding to non-satisfying assignments, the support of a is $< (n + m)$. Thus, if we set $\theta = n + m$ and $\delta = 2^{-n}$, asking whether $\{a\}$ is a probabilistic frequent itemset tells us whether F is satisfiable. □

4 DAG-R Correlated Probabilistic Database Model

4.1 Overview

DAG-R CPDM is generated by adding the following constraints to DAG CPDM. Given a PDB R which includes n transactions, $t_1, \ldots, t_n$, every tuple is grouped in a disjoint set of size k, where k is an integer from 1 to n. To give the intuition of our model we choose $k \leq 2$, where PDB contains two correlated tuples and single

Table 2. Conditional prob. for the correlation rules that are shown in Fig. 2.

v_2	v_4	Partial correlation $p(v_4\|v_2)$	Positive correlation $p(v_4\|v_2)$	Negative correlation $p(v_4\|v_2)$
T	T	1/2	2/3	0
T	F	1/2	1/3	1
F	T	1/4	0	1
F	F	3/4	1	0

tuple that is independent from others. In the correlation rules each component can only come from the tuples that are in the same group and they can only include the same items. The following example further illustrates the model.

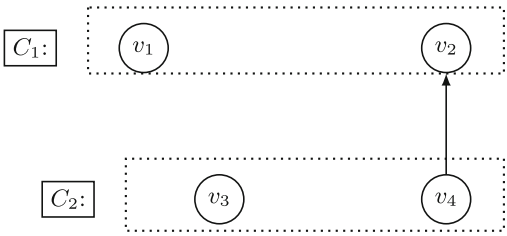


Fig. 2. Correlation rules on the PDB that is shown in Fig. 1a.

Example 2. Given the PDB in Fig. 1a, correlation rule in Fig. 2 , and conditional probabilities in Table 2, we compute all possible worlds under the conditional probability cases that are given in Subsect. 6.2 and the independent model. The complete set of possible worlds are shown in Table 3. The probability value of $\Pr(W_1)$, under partial correlation case, is computed as follows.
$\Pr(W_1) = \Pr(v_1 = T, v_2 = F, v_3 = F, v_4 = F) = \Pr(v_4 = F | v_2 = F) \times \Pr(v_2 = F) \times \Pr(v_1 = T) \times \Pr(v_3 = F) = 3/4 \times 0.8 \times 1 \times 0.6 = 0.36$
where $\{v_1 = t_A.Game, v_2 = t_A.Music, v_3 = t_B.Video, v_4 = t_B.Music\}$.

Table 3. Possible worlds for Fig. 1a under three correlation cases and independent model.

	Possible world	$p_{partial}$	$p_{positive}$	$p_{negative}$	$p_{independent}$
W_1	$\{T, F, F, F\}$	0.36	0.48	0	0.144
W_2	$\{T, F, T, F\}$	0.24	0.32	0	0.096
W_3	$\{T, F, F, T\}$	0.12	0	0.48	0.0.336
W_4	$\{T, T, F, F\}$	0.06	0.04	0.12	0.036
W_5	$\{T, T, F, T\}$	0.06	0.08	0	0.084
W_6	$\{T, T, T, F\}$	0.04	0.08/3	0.08	0.024
W_7	$\{T, F, T, T\}$	0.08	0	0.32	0.224
W_8	$\{T, T, T, T\}$	0.04	0.16/3	0	0.056

4.2 Exact Probabilistic Frequent Algorithm

The first dynamic programming based algorithm to compute the frequent probability of an itemset was proposed by Bernecker et al. [3]. To efficiently calculate the frequent probability of correlated itemsets we extend dynamic programming

based Apriori algorithm that was proposed by them. Similar to their algorithm to compute exact frequentness probability, our algorithm first calculates the frequentness probability of each itemset. Before giving recursive formulas to compute frequent probability, we define $\Pr_{i,j}(X)$. It stands for probability value that the itemset X appears i times among the first j transactions in the given correlated PDB. On a correlated PDB, where there are only two correlated tuples and independent tuples, the following recursive relationships can be defined.

$$\Pr_{i,j}(X) = \Pr_{i-1,j-1}(X) \times \Pr(X \subseteq t_j) + \Pr_{i,j-1}(X) \times (1 - \Pr(X \subseteq t_j)) \quad (2)$$

$$\Pr_{i,j}(X) = \Pr_{i-1,j-1}(X) \times \Pr(X \subseteq (t_j|\bar{t}_k)) + \Pr_{i,j-1}(X) \times \Pr(X \subseteq (\bar{t}_j|t_k)) \quad (3)$$

$$\Pr_{i,j}(X) = \Pr_{i-1,j-1}(X) \times \Pr(X \subseteq (t_j|t_k)) + \Pr_{i,j-1}(X) \times \Pr(X \subseteq (\bar{t}_j|t_k)) \quad (4)$$

$$\text{Base Cases} \begin{cases} \Pr_{i,j}(X) = 1, & \text{if } i = j = 1 \\ \Pr_{i,j}(X) = 0, & \text{if } i > 0, j = 0 \\ \Pr_{i,j}(X) = \Pr_{i,j-1}(X) \times (1 - \Pr(X \subseteq t_j)), & \text{if } i = 0, j > 0\} \end{cases}$$

where $\Pr(X \subseteq t_j)$ shows probability value that the itemset X exists in transaction t_j. Also, in above equations t_j and t_k are two correlated transactions, and $\bar{t}_j$ and $\bar{t}_k$ denotes transaction that does not occur, and $\Pr(X \subseteq (t_j|t_k))$ shows probability value that the itemset X exists in $t_j|t_k$ intersection. As a correlated PDB may include both correlated transactions and independent transactions the above equations apply accordingly. The base cases apply to both correlated and independent transactions, the Eq. 2 is used for independent transactions, and 3 and 4 are used for correlated transactions.

By using the above equations probability value of θ transactions that includes the itemset X can be calculated. This probability value is called frequentness probability of itemset X and denoted as $\Pr_{\theta,n}(X)$ where n is the number of transactions. To efficiently calculate $\Pr_{\theta,n}(X)$ value for each (i, j) pairs, j only runs up to $n-\theta+i$. Thus, large j values are excluded from $\Pr_{\theta,n}(X)$ computation.

Exact probabilistic frequent algorithm uses the dynamic programming method to compute frequent probability of each itemset and Apriori framework [1] to compute all probabilistic frequent itemsets. The time complexity of the dynamic programming computation of each itemsets is $\mathcal{O}(n^2 \times \theta)$.

5 Related Work

Correlations on PDBs has been considered in the following studies.

Tuple Correlations. San et al. [8] introduced correlations between tuples on probabilistic databases. A probabilistic graphical model "factored representation" was used for modeling correlations. Query processing was discussed on probabilistic databases that is represented by the model. They also described optimization to query processing over probabilistic databases.

x-Tuple Model. Benjelloun et al. [2] proposed the X-tuple model, which they used in the context of answering top-k queries on PDBs. Applied to uFIM, the

X-tuple model would say that the presence of one item in a transaction in a possible world rules out the presence of another item in a (possibly different) transaction in the same possible world. Such correlations can be captured in our DAG-R CPDB model.

Linear Correlations. Tong et al. [11] proposed *linear correlation* in uFIM. They argued that linear correlation is suitable for modeling dependencies in sensor data, which is correlated according to the spatial closeness of the sensors. However, linear correlation is somewhat limited in its expressive power. As it is not always possible to set linear correlation between any two variables in a way that they have linear correlation between them, but independent of other variables.

Definition 3 *(Property of Linear Correlation [6]). Given n Bernoulli random variables,* $y_1, y_2, \cdots, y_n$*, they have no second or higher order correlations if and only if Eq. 5 holds.*

$$\frac{p(y_1 = v_1, \ldots, y_n = v_n)}{p(y_1 = v_1) * \ldots, p(b_n = v_n)} = \sum_{1 \le i \le j \le n} \frac{p(y_i = v_i, y_j = v_j)}{p(y_i = v_i) * p(y_j = v_j)} - \frac{n(n+1)}{2} + 1 \tag{5}$$

where $v_l = 0$ *or* 1 *for* $l \in [1, n]$

To clarify our claim we give Example 3.

Example 3. Given three Bernoulli variables $b_1 = 0.3$, $b_2 = 0.7$, and $b_3 = 0.79$, the Pearson correlation coefficients between these pairs of variables $\rho_{1,2} = 0$, $\rho_{1,3} = 0.3375$, and $\rho_{2,3} = 0.7875$, we want to show that property of correlation coefficient which is given by Eq. 5 does not hold for these variables.

To solve Eq. 5 for different combinations of b_1, b_2, and b_3, we first compute the two sub parts of the equation $p(b_i = v_i, b_j = v_j)$ and $p(b_i = v_i, b_j = v_j)/p(b_i = v_i) * p(b_j = v_j)$. By placing back these probability values we compute the probability value of each $p(b_1 = v_1, \ldots, b_n = v_n)/p(b_1 = v_1) * \ldots, p(b_n = v_n)$. Finally we multiply the each of these equations by its denominator, and find the possible world probabilities that are shown in Table 4. The probability value of world W_2 comes out -0.0441 which is not a valid probability value. This completes our clarification.

Table 4. A subset of possible worlds for given Bernoulli variables in Example 3.

	Possible world	Probability
W_1	$\{b_1 = 1, b_2 = 1, b_3 = 1\}$	0.2541
W_2	$\{b_1 = 1, b_2 = 1, b_3 = 0\}$	-0.0441
$\ldots$	$\ldots$	$\ldots$
W_8	$\{b_1 = 0, b_2 = 0, b_3 = 0\}$	0.1659

6 Experimental Study

In this section, we report experimental results on comparison of DAG-R CPDM model with the Independent model. To compare these two models, we find all probabilistic frequent itemsets under both models by using the exact probabilistic frequent algorithm. Then, the similarity of sets of frequent items that found under both models compared by usingby using Jaccard index[2].

6.1 Experimental Setup

We run our experiments on a computer that has Intel(R) Xeon(R) E5-2620 2:40 GHz processor, 16 GB main memory, running on Ubuntu Linux3:16. The apriori algorithm technique is implemented using Java programming language. Datasets that are used for experimental evaluation are taken from Frequent Itemset Mining Dataset Repository(fimi) fimi.ua.ac.be/data/. The chosen datasets are Mushroom, Chess, Retail, and T10I4D100K, which are not probabilistic datasets. These datasets include transactions with varying number of items, where the chess and mushroom datasets has fixed number of items per transactions. Also, mushroom and chess data sets share higher number of similar items between their transactions. Whereas, T10I4D100K and Retail datasets do not share many similar items between their transactions and they are less intense than Chess and Mushroom. The characteristics of these datasets are shown in Table 5.

Table 5. Characteristic of datasets

Dataset	Number of items	Number of transactions	Average length
Chess	75	3196	37
Mushroom	119	8124	23
T10I4D100K	870	100000	10
Retail	16470	88162	10

Considering the size of the datasets and their intensity, we set a default value for each parameter differently on each dataset. This was because a default value for one dataset was not meaningful for the other one. The θ and δ values that we used in our experiments are shown in Table 6.

6.2 Correlation Types

Based on the conditional probability values we can analyze the dependencies between the tuples under the following three cases. When one of the two or more customers make a buying decision, other customer(s) will make a buying decision based on it. If the second decision maker makes the same buying decision

[2] $Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where A and B arenot empty and $0 \leq Jaccard(A, B) \leq 1$.

Table 6. θ and δ threshold for Chess, Mushroom, T10I4D100K, and Retail datasets.

Dataset	Positive correlation		Negative correlation		Partial correlation	
	θ	δ	θ	δ	θ	δ
Chess	30% to 50%	0.3	5% to 50%	0.3	5% to 50%	0.3
Mushroom	1% to 30%	0.3	1% to 20%	0.3	1% to 15%	0.3
T10I4D100K	0.015% to 0.060%	0.3	0.005% to 0.060%	0.3	0.003% to 0.020%	0.3
Retail	0.01% to 1%	0.1	0.01% to 1%	0.1	0.01% to 0.09%	0.1

as the first one, we call it *positive correlation*, while if the second one makes an opposite buying decision, it is called *negative correlation* and finally if the first one's decision does not directly influence the second one's decision we call it *partial correlation*.

6.3 Correlated Probabilistic Dataset Generation

Correlated probabilistic datasets are generated in the following two steps. In the first step, we group transactions that share the highest number of same items by using Jaccard index. To prevent complexity, we restrict our correlation rules to be in length two. The pair-up process starts with picking one transaction and scanning the database for finding the most similar transaction to it. The transactions that share similar items copied into a new file, with one transaction is adjacent to another. This process recursively continues until all transactions are paired up with one of the other transactions. During this process, every transaction is used only once. Secondly, we go through each correlation rule and add a probability value to each item of the transactions from $[0.01, 0.99]$ except the common items. For the common items probability values are given according to each case that is described in Subsect. 6.2. For example, under the positive correlation, common items were given same randomly chosen probability value of p from $[0.01, 0.99]$.

6.4 Experimental Evaluation

Minimum support threshold (θ) and frequentness probability threshold (δ) are two parameters that influence the number of frequent itemsets. We test their influence by keeping one of them at a fixed value and changing the value of the other one. For each pair of θ and δ, we compare the number of frequent itemsets that obtained from both DAG-R CPDM and independent model. Upon the frequent itemsets are obtained, Jaccard index was used to compare the similarity of sets of itemsets. We choose the minimum length of the itemsets to be three, to apply Jaccard index. This is because sets of itemsets that has length one and length two have most of the common items, so almost all sets of the itemsets had high Jaccard similarity values between them.

The common observation that we got from our experiments is that the Jaccard index values are getting closer to zero (or become zero) when the length of

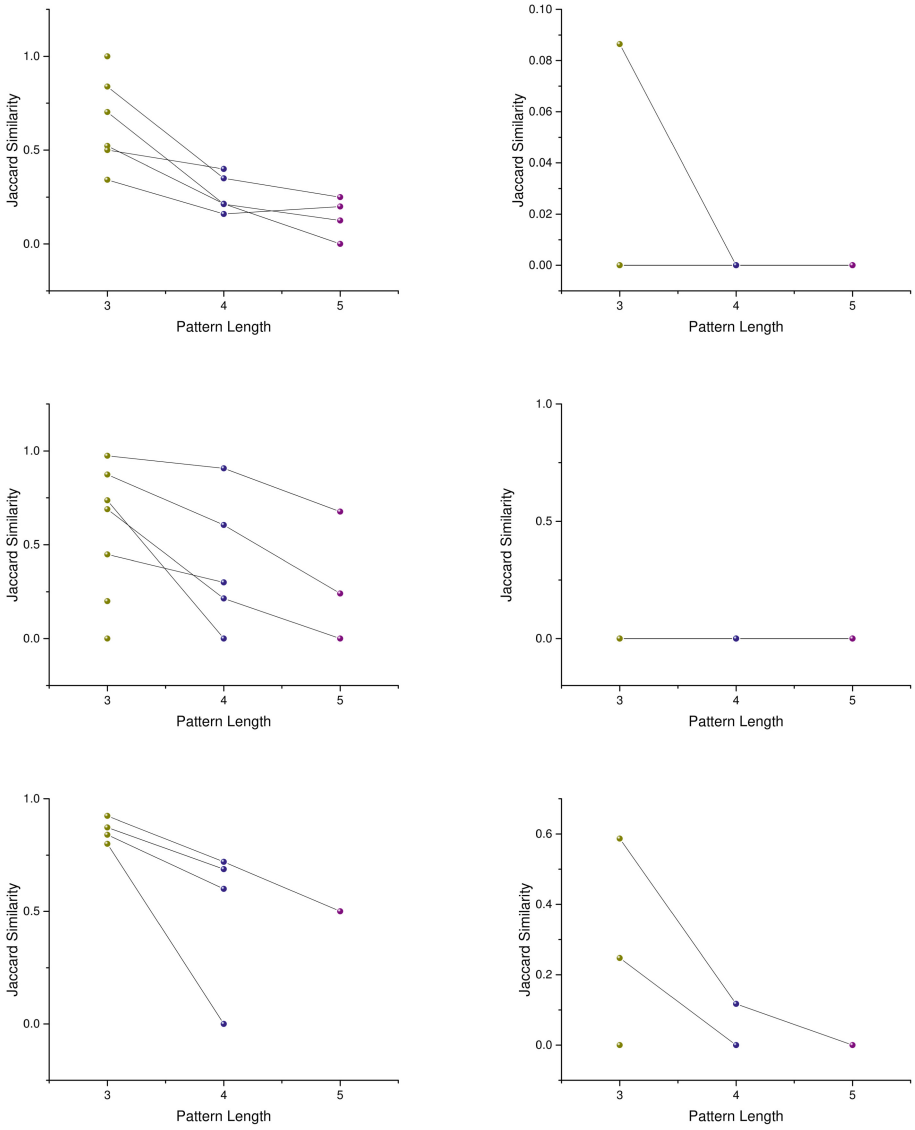


Fig. 3. Jaccard index values vs varying θ for negative correlation, positive correlation and partial correlation respectively on Retail dataset (first column) and Chess dataset(second column).

the itemsets are getting bigger. This was because the number of frequent itemsets that we find under DAG-R CPDM was greater than or equal to the number of frequent itemsets that we found under the independent model. This result supports our claim that assuming independence would cause incorrect results, as in this case, we may miss some of the frequent itemsets.

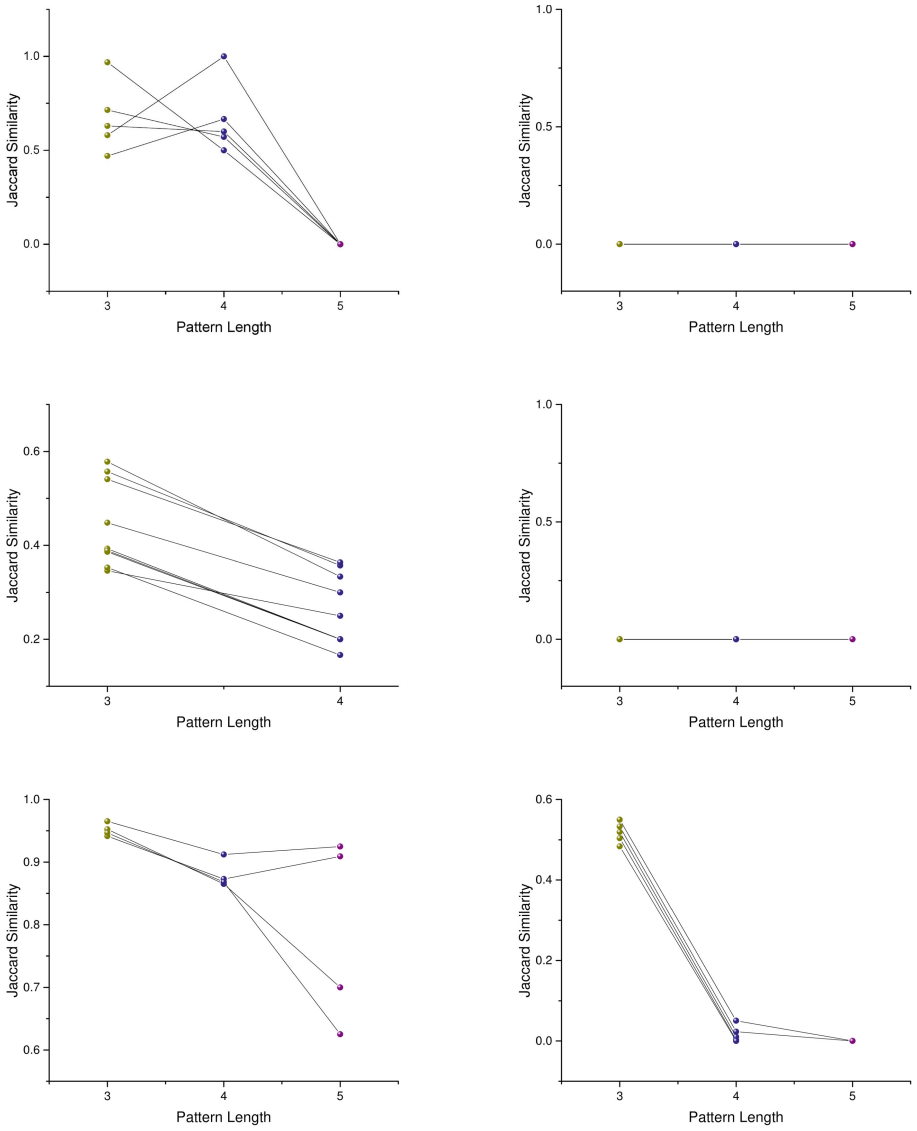


Fig. 4. Jaccard index values vs varying δ for negative correlation, positive correlation and partial correlation respectively on Retail dataset (first column) and Chess dataset (second column).

In our experiments, we were able to capture Jaccard index values up to length ten itemsets for Mushroom and Chess data sets. Whereas, for Retail and T10I4D100K we could not capture any Jaccard index values above length six. This was because the first two datasets are denser and share more similar items between their transaction, i.e., more correlated. Due to the difficulty of illustration, we only show Jaccard index values up to length five on the graphs.

Experimental results that we found for Retail dataset were close to the results of the T10I4D100K dataset, and similarly, experimental results of Chess dataset were close to Mushroom dataset' results. That is why we only show results for Retail and Chess datasets.

Effect of Minimum Support Threshold. Figure 3 shows Jaccard index values between sets of itemsets with respects to θ in Retail and Chess datasets. As it can be seen from graphs, the Jaccard index values are lower when the length of the itemsets increased. Whatsmore, for Chess dataset, Jaccard index values stay at zero for varying θ values. This is because under independent model we were not able to find any frequent itemsets w.r.t. high θ values.

Effect of Frequentness Probability Threshold. Figure 4 shows Jaccard index values between sets of itemsets w.r.t. δ. The Jaccard index values are close to zero, its is because DAG-R CPDM was able to find the higher number of frequent itemsets than the independent model. Especially for the Chess dataset, we were able to find large number of frequent itemsets under high minimum support and varying frequentness probabilities, which were not captured by the independent model.

7 Conclusion

In this study, we have shown that the DAG-R correlated probabilistic database model allows us to capture dependencies between transactions. Experimental results show that this model is able to find frequent itemsets that can not be caught by independent model. In addition, if there is no correlation rules defined this model will also find independent frequent itemsets. On the other hand, mining frequent itemsets with this model left us in an NP-complete problem when there are multiple dependencies between different database items. A solution to this problem was limiting the number of transactions in each correlation rules and excluding these transactions from others. Considering that people in real world are significantly influenced by limited number of people around them, these restrictions that we applied to the model should not affect it's applicability to real world data.

References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, vol. 1215, pp. 487–499 (1994)
2. Benjelloun, O.,Sarma, A.D., Halevy, A., Widom, J.: ULDBs: databases with uncertainty and lineage. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 953–964. VLDB Endowment (2006)
3. Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., Zuefle, A.: Probabilistic frequent itemset mining in uncertain databases. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 119–128. ACM (2009)

4. Chui, C.-K., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 47–58. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71701-0_8
5. Gupta, R., Sarawagi, S.: Creating probabilistic databases from information extraction models. In: Proceedings of the International Conference on Very Large Data Bases, vol. 32, pp. 965. Citeseer (2006)
6. Lancaster, H.O., Seneta, E.: Chi-square distribution. Wiley Online Library (1969)
7. Li, Y., Bailey, J., Kulik, L., Pei, J.: Efficient matching of substrings in uncertain sequences. In: Proceedings of the 2014 SIAM International Conference on Data Mining, pp. 767–775. SIAM (2014)
8. Sen, P., Deshpande, A.: Representing and querying correlated tuples in probabilistic databases. In: IEEE 23rd International Conference on Data Engineering 2007, ICDE 2007, pp. 596–605. IEEE (2007)
9. Suciu, D., Olteanu, D., Ré, C., Koch, C.: Probabilistic databases. Synth. Lect. Data Manag. **3**(2), 1–180 (2011)
10. Tong, W., Leung, C.K., Liu, D., Yu, J.: Probabilistic frequent pattern mining by puh-mine. In: Cheng, R., Cui, B., Zhang, Z., Cai, R., Xu, J. (eds.) APWeb 2015. LNCS, vol. 9313, pp. 768–780. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25255-1_63
11. Tong, Y.-X., Chen, L., She, J.: Mining frequent itemsets in correlated uncertain databases. J. Comput. Sci. Technol. **30**(4), 696–712 (2015)
12. Xie, D., Qin, Y., Sheng, Q.Z., Xu, Y.: Managing uncertainties in RFID applications-a survey. In: 2014 IEEE 11th International Conference on e-Business Engineering (ICEBE), pp. 220–225. IEEE (2014)
13. Zhang, Q., Li, F., Yi, K.: Finding frequent items in probabilistic data. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 819–832. ACM (2008)

Leveraging Data Relationships to Resolve Conflicts from Disparate Data Sources

Romila Pradhan(✉), Walid G. Aref, and Sunil Prabhakar

Purdue University, West Lafayette, IN 47907, USA
{rpradhan,aref,sunil}@cs.purdue.edu

Abstract. Recently, a number of data fusion systems have been proposed that offer conflict resolution as a mechanism to integrate conflicting data from multiple information providers. State-of-the-art data fusion systems largely consider claims for a data item to be unrelated to each other. In many domains, however, the observed claims are often related to each other through various entity-relationships. We propose a formalism to express entity-relationships among claims of data items and design a framework to integrate the data relationships with existing data fusion models to improve the effectiveness of fusion. We conducted an experimental evaluation on real-world data, and show that the performance of fusion was significantly improved with the integration of data relationships by (a) generating meaningful correctness probabilities for claims of data items, and (b) ensuring that the multiple correct claims output by the fusion models were consistent with each other. Our approach outperforms state-of-the-art algorithms that consider the presence of relationships over claims of data items.

1 Introduction

With the advent of the collaborative web, while innumerable data providers furnish increasing amounts of information on diverse data items, often there is little to no restraint on the quality of data from different providers. Data sources often provide conflicting information either unknowingly (e.g., failing to furnish updated data, making errors during data collection, copying from other sources) or deliberately (e.g., to mislead facts). A number of data fusion techniques have been proposed [1] to resolve data discrepancies from disparate sources and present high-quality integrated data to users. Recently, [2,3] studied the problem of dependence among sources in the context of data fusion whereas [4,5] studied the interdependence among data items in the fusion of spatial and temporal data. However, the space of existing associations between claims of data items has largely been unexplored. Failing to acknowledge these relationships has been observed to account for as much as 35% of false negatives in data fusion tasks [6]. The rich space of relationships among claims of data items makes it challenging to distinguish correct from incorrect information as illustrated next.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 99–115, 2018.
https://doi.org/10.1007/978-3-319-98812-2_7

Table 1. Table shows five websites providing information about music genres of four songs. Correct claims are marked with a (*).

ID	Data item	S_1	S_2	S_3	S_4	S_5
O_1	Silent night		*Christmas*	*Pop**	*Pop/Rock**	
O_2	Feel it still	*Pop**	{*Alt Pop Rock**, *Rap*}	*Rock**	*Pop/Rock**	*Pop**
O_3	Perfect		*Pop**	*Classical*	*Pop/Rock**	*Classical*
O_4	Unforgettable	*Rap**	{*Pop, Alt R&B**}	*Classical*	*Hip Hop**	

Example 1. Consider an example of information provided by five websites on music genres of certain songs (Table 1). Sources provide conflicting information for the same data item, e.g., S_2 provides Christmas as the genre for song Silent Night whereas S_3 claims it to be Pop and S_4 provides Pop/Rock as the genre.

Claims for data items exhibit various entity-relationships: (a) Sometimes, claims are *hierarchically* related, e.g., Pop/Rock is a sub-genre of genres Pop and Rock whereas Alt R&B has stylistic origins in Hip Hop; (b) a claim may be referred to by different names, e.g., in the context of music, Hip Hop and Rap are widely considered to agree with each other; (c) claims may be mutually exclusive to other claims. For example, the song Unforgettable may not be simultaneously of the Classical and the Hip Hop genres. Note that entity-relationships among claims can be obtained from domain-specific databases (e.g., structured vocabulary input [7], map databases) and general purpose knowledge bases [8,9]. (The relationships among claims for this example have been obtained from DBpedia [9] and AllMusic[1], the popular online music guide.)

Single-truth data fusion models [2,10] mostly regard claims to be mutually exclusive while some consider implications (or similarities) among the various observations. The approaches adopt ad hoc measures, such as string edit distance, difference between numerical values, and Jaccard similarity, to identify whether or not one claim implies another. These measures, however, may not be directly applicable to data that exhibit relationship semantics different from notions of implications addressed in prior work, e.g., when claims are real-world entities related to each other beyond string edit-distance. On the other hand, multi-truth fusion models [3,11] completely disregard the existence of relationships among claims of data items. Implications between observations may offer completely new scenarios in the multi-truth setting, e.g., integrity constraints may mandate that multiple true claims be associated to each other.

Furthermore, the correctness probabilities produced by different data fusion models often do not reflect the true likelihood of a claim being true: without any integrity constraints, a data fusion model may generate correctness probabilities such that for the song Perfect, sub-genre Pop/Rock rather than genre Pop has a higher probability of being correct. However, since the latter is a broader genre,

[1] www.allmusic.com.

one would expect it more likely to be true. Existing data fusion models do not account for these kinds of constraints on the correctness probabilities of claims.

Given the knowledge of how different music genres are related to each other, a data fusion system that considers Pop and Pop/Rock to be distinct genres (for the song Perfect) would benefit from the knowledge by re-evaluating the correctness probabilities of these claims and by reconsidering claims provided by sources S_2 and S_4 to improve the output of fusion on other data items. There are, however, certain challenges in integrating the domain knowledge information on entity-relationships among claims with the data fusion process. First, there can be permutations of agreement or disagreement among sources at different *granularities* of information. For example, sources may: (a) agree on a broader concept but disagree on specifics, (b) agree on a specific concept and disagree on broader ones, or (c) may not reach a consensus at any granularity. A naïve solution will gather evidence for and resolve the 'general' claims; however, the downside to the approach is that while we gain confidence about broader claims, no additional evidence is obtained on the correctness of specific claims. Second, existing data fusion models vary widely in their underlying conflict resolution mechanisms (e.g., Bayesian-based, optimization-based, probabilistic-graphical-model-based). We need a way to represent the data relationships that facilitates seamlessly integrating it with the various fusion models. To address the aforementioned issues, we require principled strategies to represent the domain knowledge information on relationships among claims and leverage it effectively to jointly assess data sources and infer correctness probabilities of claims.

In this paper, we address the problem of integrating entity-relationships among claims with data fusion process to improve the effectiveness of existing data fusion models. Our main contributions can be summarized as follows:

- We propose to represent the knowledge of data relationships among claims in the form of an arbitrary directed graph. We outline pre-processing steps for effective representation and efficient traversal of the graph (Sect. 4).
- We propose an approach to integrate the directed graph of data relationships with existing data fusion models and propose an algorithm to leverage the graph to generate consistent correct claims for each data item (Sect. 5).
- Our experimental evaluation on real-world data shows the applicability of our approach to a wide range of data fusion models and demonstrates that incorporating the domain knowledge of entity-relationships among claims can significantly improve fusion results (Sect. 6).

2 Related Work

Data Fusion. The problem of conflict resolution as a way to integrate conflicting data from a multitude of data sources has been extensively studied, and a number of data fusion systems have been proposed in the past [1].

The present work provides a general framework to effectively integrate relationships among claims of data items with existing data fusion models.

Leveraging Data Correlations. The problem of dependencies and correlations among data sources has been studied in the context of data fusion [2,3] whereas correlations between data items have been explored during the fusion of spatial and temporal data [4,5].

While the hierarchical structure of relationships among object labels has been studied extensively in the past, especially in the area of image annotation [12] and classification [13], it has not been exploited fully in data fusion. Few single-truth data fusion systems (that assume each data item to have a single correct claim) have found the approach of considering implications or similarities between claims to improve the effectiveness of fusion [2,10]; the adopted techniques, however, are limited to ad hoc similarity measures between claims. Multi-truth models [3,11], on the other hand, do not consider any associations among claims of data items.

The closest to our work is [14] that proposed using information on partial ordering among claims to discover truth from synthetically generated data and showed that considering partial ordering reduces the error-rate of source quality estimation. Their approach, however, does not capture relations other than partial ordering, e.g., it does not address representation of relations among claims that are equivalent to each other or are mutually exclusive. Moreover, in a bid to limit overestimation, the approach does not take the partial order into account for evaluating source metrics, and considers it partially in determining correct claims of data items resulting in a low overall recall for fusion.

Extracting Entity-Relationships. The present work does not focus on modeling the entity-relationships for integration with data fusion. With the undeniable success of large-scale knowledge bases [8,9] and the ongoing research on learning entity-relationships [15,16], our framework relies on knowledge bases and domain-specific databases to extract the relationships among claims of data items. The extracted relations are then fed into the data fusion framework to improve the effectiveness of fusion.

3 Problem Formulation

We consider database instance $\mathcal{D}$, data fusion model $\mathcal{F}$ and binary relation $\mathcal{R}$ denoting the entity-relationships among claims of data items in $\mathcal{D}$, and formulate the problem of leveraging relation $\mathcal{R}$ to improve the effectiveness of fusion.

Data Model. Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a set of sources that provide claims about data items in set $\mathcal{O} = \{O_1, \ldots, O_m\}$. For a particular data item, say O_i, $S^i = \{S^i_1, S^i_2, \ldots\}$ denotes the ordered list of sources that provide claims $\psi^i = \{\psi^i_1, \psi^i_2, \ldots\}$ about O_i, where source S^i_j provides claim ψ^i_j. The set of unique claims of O_i is denoted by $V_i = \mathit{distinct}(\psi^i) = \{v^1_i, \ldots, v^{|V_i|}_i\}$. The set of sources that provide claim $v \in V_i$ is represented by $S^i(v) \subseteq S^i$, and the set of claims that S_j provides for O_i is denoted by $V_i(S_j) \in V_i$. We represent the observations and distinct claims of data items in $\mathcal{O}$ by $\Psi = \{\psi^1, \ldots, \psi^{|\mathcal{O}|}\}$ and $V = \{V_1, \ldots, V_{|\mathcal{O}|}\}$, respectively.

Example 2. Consider data item O_2 in the example presented in Table 1. $\psi^2 = \{\mathsf{Pop}, \mathsf{Alt\ Pop\ Rock}, \mathsf{Rap}, \mathsf{Rock}, \mathsf{Pop/Rock}, \mathsf{Pop}\}$ is the ordered list of claims made by sources in $S^2 = \{S_1, S_2, S_2, S_3, S_4, S_5\}$, where source S_3 provides claim Rock. Also, $V_2 = \{\mathsf{Pop}, \mathsf{Alt\ Pop\ Rock}, \mathsf{Rap}, \mathsf{Rock}, \mathsf{Pop/Rock}\}$. For data item O_2, the set of sources for claim Pop is denoted by $S^2(\mathsf{Pop}) = \{S_1, S_5\}$.

Definition 1. *A database $\mathcal{D}$ is a tuple $\langle \mathcal{O}, \mathcal{S}, \Psi, V \rangle$, where $\mathcal{O}$ is the set of data items, $\mathcal{S}$ is the set of sources, $V = \{V_1, \ldots, V_{|\mathcal{O}|}\}$ is the set of claims, and $\Psi = \{\psi^1, \ldots, \psi^{|\mathcal{O}|}\}$ is the set of observations for all data items.*

Definition 2. *A data fusion system $\mathcal{F}$ is a function that takes database $\mathcal{D}$ as input and outputs a set of probability assignments P denoting correctness probabilities of claims and source quality measures $Q^{\mathcal{F}}$:*

$$\mathcal{F} : \mathcal{D} \rightarrow \langle P, Q^{\mathcal{F}} \rangle$$

where $\forall O_i \in \mathcal{O}$, $P(v_i^k) = p_i^k \in [0, 1]$ is the correctness of claim v_i^k, i.e., the probability that claim $v_i^k \in V_i$ is correct and $\forall S_j \in \mathcal{S}$, $Q_j^{\mathcal{F}}$ is a vector indicating the quality of source S_j.

Definition 3. *A binary relation $\mathcal{R} \subseteq V \times V$ denotes the entity-relationships among claims $V = \{V_1, \ldots, V_{|O|}\}$ of data items in $\mathcal{O}$.*

Problem Statement. It is required to develop a *relation-aware* data fusion framework, denoted by $\mathcal{F}_{\mathcal{G}}$, that integrates data fusion model $\mathcal{F}$ with relation $\mathcal{R}$ to infer the correctness probabilities of claims in database $\mathcal{D}$.

4 Exploring Entity-Relationships

In this section, we review the various entity-relationships existing between claims of data items and propose a formalism to express the prior domain knowledge of entity-relationships among claims.

4.1 Observations

As an extension to existing relationships among real-world entities, we observe subsumption, overlaps, equivalence and disjointedness among claims of data items (also detailed in Example 1). In the following, we provide an intuition of what these relationships mean in the context of correctness of claims:

Subsumption/Overlaps. A claim may be part of one or more claims, e.g., Pop and Rock, as music genres, are generalization of the Pop/Rock genre. Any source that provides Pop/Rock definitely agrees with the Pop and Rock genres. We say that genre Pop/Rock *implies* or *supports* genres Pop and Rock.

Equivalence. Real-world entities may be referred to differently by different sources and contexts, e.g., Hip Hop music is referred to as Rap in some cultures and contexts. Therefore, any source that provides Hip Hop as a genre agrees with Rap and vice versa. The relation between such claims elicits a bidirectional implication, i.e., both the claims imply each other.

Mutual exclusion. In certain settings, the correctness of a claim may require all other claims to be declared false. For example, a song-listing integration system may mandate that a song be either of genre Alt R&B or Classical but not both. Therefore, if Alt R&B is considered the correct genre for data item O_4, Classical cannot be correct and vice versa.

From these observations, we recognize two themes, namely *implication* and *mutual exclusion*, in the relationship among claims of data items. Implication summarizes subsumption, overlaps and equivalence relationships, and indicates claims that can be correct or incorrect at the same time. Mutual exclusion dictates the set of claims that cannot be simultaneously correct.

4.2 Relationship Model

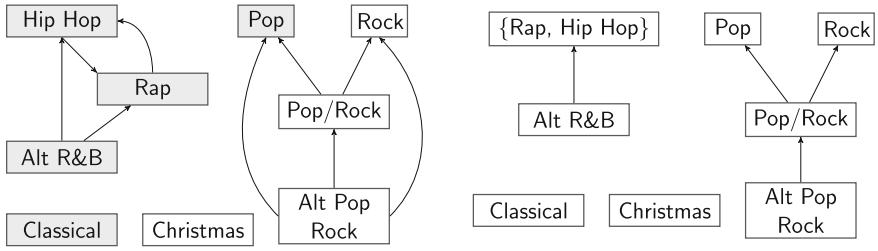
Based on these two themes, we define relation $\mathcal{R} \subseteq V \times V$ to describe implication (relationship) between two claims: that is $(u, v) \in \mathcal{R}$ if and only if u implies or supports v. We observe that $\mathcal{R}$ is reflexive, transitive and neither symmetric nor antisymmetric (because given $(u, v) \in \mathcal{R}, (v, u)$ may or may not exist in $\mathcal{R}$). Relation $\mathcal{R}$ can be represented in the form of a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = V$, i.e., vertices in $\mathcal{G}$ represent the set of distinct claims in V and edges in $\mathcal{E}$ represent the relation between claims at the corresponding vertices. $\forall (u, v) \in \mathcal{R}, \exists (u, v) \in \mathcal{E}$ denoting the fact that claim represented by vertex u supports that represented by v. In the rest of the paper, where applicable, we will use claim $v \in V$ and the vertex represented by claim $v \in \mathcal{V}$ interchangeably. Subgraph $G_i = (V_i, E_i) \subseteq \mathcal{G}$ represents the relations over claims of data item O_i.

Following standard graph notation, if $e = (u, v) \in \mathcal{E}$, then v is a parent of u and u is a child of v. If there is a path from u to v (denoted by $u \rightsquigarrow v$), then v is an *ancestor* of u and u is a descendant of v. An arbitrary directed graph thus defined captures the observed relations among claims in the following way:

Implication relation is captured by reachability among vertices. If $u \rightsquigarrow v$ in $\mathcal{G}$, then u implies or supports v. Under this definition of implication,

1. v represents *coarser* information than u and encapsulates subsumption.
2. Overlapping claims have a common descendant. Formally, u overlaps with v if there exists w such that $w \rightsquigarrow u$ and $w \rightsquigarrow v$.
3. If $u \rightsquigarrow v$ and $v \rightsquigarrow u$, then u and v represent equivalent claims such that $\mathcal{G}$ contains a cycle which is incident with both u and v. Equivalent claims are represented by *equivalence classes* of vertices in $\mathcal{G}$.

Mutual exclusion is expressed by identifying claims that do not have a common descendant, i.e., u and v are mutually exclusive if $\nexists w$ such that $w \rightsquigarrow u$ and $w \rightsquigarrow v$.



(a) Directed graph $\mathcal{G}$. **(b) Modified directed graph $\mathcal{G}^{\mathcal{T}}$.**

Fig. 1. Figure on left shows the directed graph $\mathcal{G}$ of entity-relationships among claims of data items as shown in Table 1. The shaded subgraph denotes relations between claims specific to data item O_4. Figure on right shows modified graph $\mathcal{G}^{\mathcal{T}}$ obtained as transitive reduction of the equivalent acyclic graph of $\mathcal{G}$.

A directed graph (defined as above) over the claims of a data item presents general to specific information as we move from its root (top) to leaves (bottom). When claims are not related, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be seen as a graph with claims as vertices with no edges in between, i.e., $\mathcal{E} = \emptyset$.

Example 3. Figure 1a shows the directed graph of relations over claims of data items in Table 1. Rock and Pop are overlapping claims that have a common descendant: Pop/Rock. Hip Hop and Rap are considered equivalent claims as they are on a cycle incident with both the claims. Moreover, claims Rap and Christmas are mutually exclusive because they do not have a common descendant.

Removing Redundancies. The aforementioned directed graph representation can have a large number of redundant edges and vertices as illustrated next. Consider subgraph $G_2 = (V_2, E_2) \subseteq \mathcal{G}$ consisting of claims of data item O_2. Since edge (Alt Pop Rock, Pop/Rock) $\in E_2$ and edge (Pop/Rock, Rock) $\in E_2$, by transitivity, Alt Pop Rock $\rightsquigarrow$ Rock causing edge (Alt Pop Rock, Rock)$\in E_2$ to be redundant. Furthermore, in the subgraph $G_4 = (V_4, E_4) \subseteq \mathcal{G}$ of claims of data item O_4, claims Hip Hop and Rap are in the same equivalence class and therefore, can be represented by a single vertex.

We process graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in the following two steps to achieve a concise representation that facilitates effective summarization and efficient navigation:

1. **Redundant Vertices.** We remove redundant vertices in $\mathcal{V}$ by forming the equivalent acyclic graph [17] of $\mathcal{G}$, denoted by $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$. Vertices in $\mathcal{G}^*$ represent equivalence classes in $\mathcal{G}$ and edges in $\mathcal{G}^*$ represent edges between the equivalence classes. $\mathcal{G}^*$ can be obtained by identifying strongly connected components [18] of $\mathcal{G}$. Consider vertices $u, v \in \mathcal{V}$. Let u^* and v^* respectively represent the equivalence classes for claims u and v in $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$. For $u^* \neq v^*$, if $\exists (u, v) \in \mathcal{E}$, then edge $(u^*, v^*) \in \mathcal{E}^*$. Note that $\mathcal{G}^*$ may still have redundant edges because of the transitivity property.

2. **Redundant Edges.** We identify the unique transitive reduction [17] of $\mathcal{G}^*$, denoted by $\mathcal{G}^{\mathcal{T}} = (\mathcal{V}^*, \mathcal{E}^{\mathcal{T}}) \subseteq \mathcal{G}^*$. $\mathcal{G}^{\mathcal{T}}$ has no redundant edge, i.e., for $u, v \in \mathcal{V}^*$, if v is not a parent of u and $u \rightsquigarrow v$, then edge $(u, v) \notin \mathcal{E}^{\mathcal{T}}$. Transitive reduction $\mathcal{G}^{\mathcal{T}}$ has the fewest possible edges and has the same reachability relation as $\mathcal{G}^*$.

Subgraph $G_i^T = (V_i^*, E_i^*) \subseteq \mathcal{G}^{\mathcal{T}}$ represents the transitive reduction of G_i. Figure 1b shows the graph obtained after processing directed graph in Fig. 1a.

Complexity Analysis. Equivalent acyclic graph $\mathcal{G}^*$ is obtained in $O(|\mathcal{V}| + |\mathcal{E}|)$ time [18] whereas transitive reduction $\mathcal{G}^{\mathcal{T}}$ can be derived in $O(|\mathcal{V}|^{\beta})$ steps [17] where $\beta \geq 2$. Note that processing directed graph G_i depends only on the number of distinct claims for data item O_i (which is usually not very large) and not on the number of sources that provide information on the item.

In the rest of this paper, we use $\mathcal{G}$ to represent the modified directed graph representation $\mathcal{G}^{\mathcal{T}}$ and G_i to denote G_i^T.

Supporting and Supported Claims. To integrate directed graph $\mathcal{G}$ with existing data fusion models, we need to identify the following two sets of claims for each claim $v \in V_i$: (a) set of claims in V_i that support v, denoted by $\delta(v, G_i)$; and (b) set of claims in V_i that v supports, denoted by $\alpha(v, G_i)$. After identifying the vertex or equivalence class in G_i that v belongs to, we add claims in the equivalence class and claims that are its descendants in G_i to $\delta(v, G_i)$, and add claims in the equivalence class and claims that are its ancestors in G_i to $\alpha(v, G_i)$. The notion of supporting and supported claims will be used in Sect. 5.1 to estimate source qualities and correctness of claims.

5 Integration with Data Fusion

Given the entity-relationships among claims as described in $\mathcal{G}$ (obtained in Sect. 4), in this section we outline the steps for leveraging $\mathcal{G}$ to resolve conflicts during integration of data from multiple sources. We first describe how existing data fusion models can be modified in the presence of $\mathcal{G}$ and then discuss how to utilize $\mathcal{G}$ to determine correct claims for data items.

5.1 Revised Data Fusion Methodology

To determine which of the provided claims are correct and which incorrect, state-of-the-art data fusion models [2,3,10] consider sources to play a pivotal role and usually function in two steps: first, obtain source quality estimates; second, compute the correctness of claims based on the computed source qualities. Given a data fusion model $\mathcal{F}$, characterized by computations of source quality measures $Q^{\mathcal{F}}$ and correctness of claims P, we describe how to modify these two computations for $\mathcal{F}$ given directed graph $\mathcal{G}$ over claims of data items.

- **Estimating Source Quality.** Existing fusion models evaluate sources either in terms of a single measure (e.g., accuracy [2], trustworthiness [10]) or multiple measures (e.g., precision, recall, accuracy, false positive rate [3,11]).

Table 2. Claims provided or supported by source S_2.

ID	$V_i(S_2)$	$\overrightarrow{V_i}(S_2)$	Correct
O_1	Christmas	Christmas	Pop, Pop/Rock
O_2	Alt Pop Rock, Rap	Alt Pop Rock, Pop/Rock, Pop, Rock, Rap	Alt Pop Rock, Pop/Rock, Pop, Rock
O_3	Pop	Pop	Pop/Rock, Pop
O_4	Pop, Alt R& B	Pop, Alt R& B, Hip Hop, Rap	Alt R&B, Hip Hop, Rap

The quality of source S_j, denoted by $Q_j^{\mathcal{F}}$, is measured based on $V_i(S_j)$, the set of claims that S_j provides for data item $O_i \in \mathcal{O}$.
In the presence of entity-relationships among claims, a source, in addition to claims directly provided by it, also implicitly supports claims that are supported by the provided claims. Therefore, $Q_j^{\mathcal{F}}$ depends on claims in $V_i(S_j)$ and claims supported by those in $V_i(S_j)$. Given directed graph $G_i \subseteq \mathcal{G}$ for data item O_i, claim $v \in V_i(S_j)$ supports claims in $\alpha(v, G_i)$ (Sect. 4). Consequently, we replace $V_i(S_j)$ by $\overrightarrow{V_i}(S_j) = \{\alpha(v, G_i) \mid v \in V_i(S_j)\}$ in the computation of $Q_j^{\mathcal{F}}$. Clearly, $V_i(S_j) \subseteq \overrightarrow{V_i}(S_j)$.

Example 4. Consider source S_2 in Table 1. Using the modified directed graph in Fig. 1, we observe that S_2 supports claims as shown in Table 2. Note that for each data item, we only consider the modified directed subgraph over claims of that particular data item, e.g., since claim Hip Hop $\notin V_2$, we do not consider that Rap supports Hip Hop in the context of data item O_2.

Comparing Table 2 with Table 1, we observe that out of the 11 claims S_2 supports, 8 are correct resulting in a precision (fraction of claims provided that are correct) of $8/11 = 0.73$. Its recall (fraction of correct claims provided) is $8/11 = 0.73$ as it provides 8 out of the 11 listed correct claims. Note that in the absence of knowledge of relations among the claims of data items, the precision and recall of S_2 would be $3/6 = 0.5$ and $3/11 = 0.27$, respectively.

Procedure EstimateSourceQuality outlines pseudocode for estimating source quality measures given a fusion model and claim relationships. Note that when training data is available, $P(v)$ is defined for items in the training data and $Q^{\mathcal{F}}$ is computed over those items. Otherwise, $Q^{\mathcal{F}}$ is initialized to random values, and source quality and claim correctness are estimated iteratively.

- **Estimating Correctness of Claims.** The second step in data fusion models estimates the correctness of claims by utilizing the estimated source quality measures. The correctness of claim $v \in V_i$, denoted by $P(v)$, is computed in terms of the quality measures of sources in $S^i(v)$, the set of sources that provide v. Claims provided by good sources are considered more likely to be correct than those provided by poor sources.

Procedure EstimateSourceQuality

Input: Database $\mathcal{D}$, directed graph $\mathcal{G}$, fusion model $\mathcal{F}$, claim correctness P
Output: $Q^{\mathcal{F}}$, quality measures of sources
for $s \in \mathcal{S}$ **do**
  for $O_i \in \mathcal{O}$ **do**
    $\overrightarrow{V_i}(s) = \{\alpha(v, G_i) \mid v \in V_i(s)\}$
    for $v \in \overrightarrow{V_i}(s))$ **do**
      Compute $Q^{\mathcal{F}}(s)$ according to $\mathcal{F}$ based on $P(v)$

Intuitively, the correctness of claim v should depend not only on sources that provide v but also on sources that implicitly support it – the latter can be identified by identifying claims that support v. Given directed graph $G_i \subseteq \mathcal{G}$ for data item O_i, claim v is supported by claims in $\delta(v, G_i)$. In estimating the correctness of v by a particular data fusion model, we replace $S^i(v)$ by $S^i(\overrightarrow{v}) = \{S^i(u) \mid u \in \delta(v, G_i)\}$. Again, $S^i(v) \subseteq S^i(\overrightarrow{v})$. This step ensures that general claims gather greater evidence with support from specific claims and have higher correctness probabilities than them.
In the presence of directed graph G_i, instead of computing the correctness of each provided claim for data item O_i, we compute the correctness of each vertex in G_i. Doing so, we avoid having to separately estimate the correctness of equivalent claims. Procedure EstimateClaimCorrectness outlines the pseudocode for computing correctness probabilities given the knowledge of relations among claims.

Procedure EstimateClaimCorrectness

Input: Database $\mathcal{D}$, directed graph $\mathcal{G}$, fusion model $\mathcal{F}$, source measures $Q^{\mathcal{F}}$
Output: P correctness probability of claims
for $O_i \in \mathcal{O}$ **do**
  for claim $v \in V_i$ **do**
    $S^i(\overrightarrow{v}) = \{S^i(u) \mid u \in \delta(v, G_i)\}$
    for $s \in S^i(\overrightarrow{v})$ **do**
      Compute $P(v)$ according to $\mathcal{F}$ based on $Q^{\mathcal{F}}(s)$

Given observations ψ, data fusion model $\mathcal{F}$ and directed graph representation $\mathcal{G}$, as discussed above, we integrate $\mathcal{G}$ with the processes of estimating source quality measures and correctness of claims. We present the pseudocode for modifying $\mathcal{F}$ using $\mathcal{G}$ in Algorithm 1.

Iterative fusion models [2,10] randomly initialize source quality estimates and iterate over lines 1 and 2 until $Q^{\mathcal{F}}$ converges. When ground truth data is available, fusion models [3] utilize it to compute source quality estimates.

Algorithm 1. ModifyDataFusion

Input: Database $\mathcal{D}$, directed graph representation $\mathcal{G}$, data fusion model $\mathcal{F}$
Output: P correctness probabilities of claims

1 $Q^{\mathcal{F}}$ = EstimateSourceQuality($\mathcal{D}, \mathcal{G}, \mathcal{F}, P$)
2 P = EstimateClaimCorrectness($\mathcal{D}, \mathcal{G}, \mathcal{F}, Q^{\mathcal{F}}$)

5.2 Determining Correct Claims

Having obtained the correctness probabilities, single-truth fusion models will consider claim with the highest probability to be correct and multi-truth fusion models will consider claims with probability greater than a threshold (usually 0.5) to be correct. However, determining correct claims in the standard manner has certain limitations: (a) single-truth fusion models will miss multiple correct claims, and (b) multi-truth fusion models may output correct claims that are indeed constrained to be mutually exclusive.

To address the aforementioned issues, given correctness probabilities P and directed graph $\mathcal{G}$, we describe the steps to determine correct claims for data items in Algorithm 2. Lines 4–6 identify root nodes of the directed graph G_i over claims of data item O_i. Lines 8–10 consider the vertex with maximum correctness probability, `currentNode`, to be correct and add claims in `currentNode` to the list of correct claims for data item O_i. The algorithm then identifies children nodes of the selected vertex for further traversal and repeats lines 8–10 until a leaf node (i.e., vertex with no children) is reached.

Algorithm 2. DetermineCorrectClaims

Input: Directed graph representation $\mathcal{G}$, correctness probabilities P
Output: V^*, set of correct claims for data items in $\mathcal{O}$

1 **for** $O_i \in \mathcal{O}$ **do**
2 Initialization: `considerNodes` $= \emptyset$; $V_i^* = \emptyset$
3 Let $G_i = (V_i, E_i) \subseteq \mathcal{G}$ be the directed graph over claims in V_i
4 **for** vertex $V \in V_i$ **do**
5 **if** $\nexists \{(V, b) \in E_i\}$ **then**
6 `considerNodes` $=$ `considerNodes` $\cup \{V\}$ /* identify root nodes */
7 **do**
8 `currentNode` $= \underset{w \in \texttt{considerNodes}}{\text{argmax}} P(w)$
9 **for** claim $v \in$ `currentNode` **do**
10 $V_i^* = V_i^* \cup \{v\}$
11 `considerNodes` $=$ children of `currentNode`
  while ($\exists u \mid (u,$ `currentNode`$) \in E_i$)

6 Experimental Evaluation

This section presents an empirical evaluation of the proposed approach on a real-world dataset. Our objectives are: (1) to assess the effectiveness of using the knowledge of entity-relationships among claims in improving the accuracy of existing data fusion models, and (2) to compare the effectiveness of using arbitrary directed graphs against existing approaches that consider prior domain knowledge of entity-relationships among claims of data items.

Competing Methods

We evaluate the effectiveness of using the domain information on entity-relationships among claims on the following single- and multi-truth data fusion models:

Voting: Naïvely assumes correct data to be more frequent than inaccurate data and considers the most frequent claim of a data item to be correct.

TruthFinder [10]: Iteratively computes trustworthiness of sources and confidence in claims, and selects claim with the highest confidence to be correct.

ACCU [2]: Iteratively computes accuracy of sources and correctness of claims by assuming only one claim of a data item to be correct and rest incorrect.

PrecRec [3]: Computes source quality metrics assuming access to ground truth for a subset of data items and uses the estimates to determine correctness of claims. The method outputs multiple correct claims for a data item.

We further compared our approach of using arbitrary directed graphs (denoted by DG) to the partial ordering solution [14] (denoted by PO). We implemented all the algorithms in Java.

Performance Metrics

To evaluate effectiveness of the approaches, we present results according to their *precision*, *recall* and F_1*-score*. We measure the precision of an approach as the fraction of claims output by the algorithm that are indeed true. Recall is measured as the fraction of all correct claims that are output by the particular algorithm. We measure the overall performance of an approach in terms of the harmonic mean of its precision and recall, that weighs the two metrics evenly $\left(\text{i.e.}, F_1 = \frac{2.precision.recall}{precision+recall}\right)$.

% inconsistency: We use the entity-relationships among claims of data items to measure the fraction of pairs of claims considered correct by a data fusion model that are unrelated and *inconsistent* with each other.

Table 3. Effectiveness of data fusion models on Restaurants. Multi-truth fusion model PrecRec is effective in identifying correct claims but outputs claims that may be inconsistent with each other.

	Voting	TruthFinder	ACCU	PrecRec
Recall	0.210	0.243	0.251	0.919
Precision	0.758	0.874	0.904	0.835
F1	0.329	0.380	0.393	0.875
% inconsistent	-	-	-	0.146

Real-World Data

We conducted experiments on the Restaurants dataset in [19] that lists information on restaurants in New York's Manhattan area as provided by 12 sources. We observed that the locations of these restaurants are conflicting but related and, therefore, chose to determine their correct values for the snapshot of data collected on the last available date (3/12/2009).

We identified restaurants by their names and removed those that were chains: if a single source provides inconsistent claims for a restaurant, we consider it to be a chain that may have multiple locations and remove all instances of such restaurants. For example, if a source provides two neighborhoods or two street addresses for the same restaurant, we consider the possibility that it is part of a chain of restaurants. The resulting dataset had 11,589 unique restaurants (we collected ground truth for 500). It should be noted that, we assume sources to be self-consistent (i.e., a source by itself does not provide inconsistent claims) and ignore errors arising during data collection by humans and sensors.

We extracted the different granularities of locations for restaurants as provided by sources into separate claims. For example, claim "357 East 50th St, Midtown East" was broken down into claims: 357 East 50th St and Midtown East. We extracted relations among the claims using Wikipedia [2] and corroborated with DBpedia and Google Maps. Using the neighborhood definitions, we extracted relations of streets and avenues with neighborhoods. We identified ∼1% of restaurants for manual review of relations. Their claims included buildings that were represented by alternate street addresses because of the difference in data collection strategies of different sources.

As a result of inconsistencies across data sources, the resulting directed graph of relations among claims is not just a tree (as in the partial order solution [14]) but can be any arbitrary directed graph with cycles. A partial ordering solution, therefore, will not be directly applicable to resolve such conflicting data.

The Case for Consistency. To demonstrate the need for approaches that generate consistent correct claims, we run the described data fusion models (Voting, TruthFinder, ACCU and PrecRec) on Restaurants and report their performance as measured by precision, recall and F1-measure in Table 3. We observe that while

[2] https://en.wikipedia.org/wiki/List_of_Manhattan_neighborhoods.

the multi-truth model (PrecRec) is, expectedly, able to retrieve a larger fraction of correct claims, it is less accurate than the single-truth models TruthFinder and ACCU. We dig deeper into the recall of PrecRec and observe that ∼15% of pairs of claims considered correct by PrecRec are, in fact, inconsistent with each other (similar results were obtained with synthetic data). The reason for this behavior is that the model considers most of the claims to be correct but is unable to *distinguish* correct from incorrect information. Moreover, the other methods output a single true claim, and hence are inadequate for the current problem. This experiment proves that multi-truth data fusion models are not sufficient for such interrelated data, and that there is indeed a need for approaches that present consistent and accurate data to users.

Effectiveness of Using Data Relationships During Fusion. We evaluate the advantage of using the knowledge of relations among claims of data items over the effectiveness of different data fusion models. In particular, we have three goals: (a) to evaluate whether the knowledge of relations among claims improves fusion results, (b) to compare the two approaches, PO and DG, and (c) to evaluate how the different data fusion models perform with the knowledge of relations.

Table 4. Effect of integrating the entity-relationships among claims on the effectiveness of different fusion models.

	Voting		TruthFinder		ACCU		PrecRec	
	PO	DG	PO	DG	PO	DG	PO	DG
Recall	0.889	**0.950**	0.876	**0.939**	0.797	**0.940**	0.889	**0.954**
Precision	0.948	**0.951**	0.939	**0.941**	**0.954**	0.944	0.956	**0.957**
F1	0.917	**0.950**	0.906	**0.940**	0.868	**0.942**	0.921	**0.956**

We present in Table 4, the results of using DG and PO, entity-relationships among claims, in conjunction with the data fusion models. Comparing the results with Table 3, we find that leveraging data relationships results in an overall improvement in the precision, recall and F1-measure of all data fusion models. The reason for this improvement is that using the knowledge of entity-relationships among claims: (a) single-truth fusion models are converted into multi-truth models, thus retrieving more than one correct claims for each data item and resulting in higher recall, and (b) proper traversal of the graph structures results in less false positives compared to that obtained without the information on relations.

In Fig. 2, we compare how the entity-relationship models (PO and DG) fare in conjunction with different data fusion models. Since PO does not support partial orders between claims that result in graphs with cycles, to evaluate PO, we removed edges on cycles in the directed graphs. To determine correct claims in PO, we set the probability threshold, $\theta = 0.05$, i.e., claims with correctness probability higher than 0.05 are considered correct. While both approaches exhibit

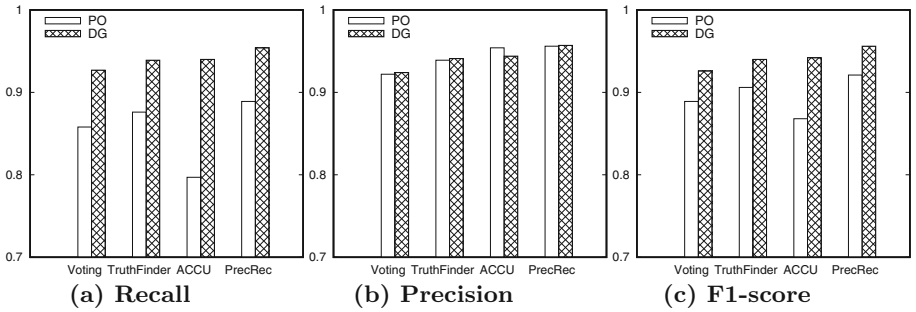


Fig. 2. Comparing relationship models PO and DG during fusion of Restaurants. For PO, we set probability threshold $\theta = 0.05$.

comparable improvement in precision, DG has consistently higher recall for corresponding data fusion models. This is because DG considers a wide range of relations existing among claims whereas PO is limited only to hierarchies and leaves out ancestors of an overlapping claim that are not reachable from the parent of the claim in question. With an increase in the value of θ, we observe that PO is able to retrieve far fewer correct claims than DG (a difference of around 20% in recall when $\theta = 0.1$ and ∼70% with $\theta = 0.3$).

It is worth mentioning how the data fusion models compare against each other in the presence of information about relations. Unsurprisingly, our best case is using DG with PrecRec, when we have access to ground truth for computing source quality measures and have all the information on relations among claims, thus outperforming the other data fusion models across all performance metrics. This is in line with earlier efforts in data fusion that emphasize upon the need for accurate initialization of source quality metrics toward obtaining superior fusion results. It is, however, interesting to note that with the knowledge of data relationships, even the most naïve data fusion technique (Voting) achieves significant improvement in precision and recall – it outperforms state-of-the-art multi-truth model PrecRec that has access to ground truth but no access to domain knowledge (comparing Voting + DG in Table 4 vs. PrecRec in Table 3).

Experiment Takeaways. (1) Leveraging the knowledge on relations among claims improves fusion results. (2) Arbitrary directed graph representation DG is more effective at identifying correct claims than partial ordering solution PO. (3) Unsupervised data fusion models (Voting, TruthFinder, ACCU) perform comparable to supervised models (PrecRec) with DG. This experiment gives rise to an important result: in the presence of domain knowledge, we may not need sophisticated models or ground truth to benefit from the domain knowledge.

7 Conclusions

In this paper, we proposed a formalism to express the prior knowledge of entity-relationships among claims of data items that enables representing a wide range

of relationship semantics existing between claims. We designed a framework to integrate the data relationships with the process of fusing conflicting data from disparate sources. We demonstrated the applicability of our approach to a number of existing fusion models, evaluated our approach against other methods that incorporate such relation information in the data, and showed that, compared to other methods, our algorithm achieves significant improvement in fusion results. The effectiveness of our approach depends on completeness of the extracted knowledge and can be improved by accounting for ambiguity in relations. Moreover, directed graphs formalize binary entity-relationships that could be improved with more expressive knowledge representation formalisms (e.g., logic-based, conceptual graphs). We plan to explore these issues in future work.

References

1. Li, Y., et al.: A survey on truth discovery. SIGKDD Explor. Newsl. **17**(2), 1–16 (2016). https://doi.org/10.1145/2897350.2897352
2. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: the role of source dependence. In: PVLDB (2009)
3. Pochampally, R., Das Sarma, A., Dong, X.L., Meliou, A., Srivastava, D.: Fusing data with correlations. In: SIGMOD (2014)
4. Meng, C., et al.: Truth discovery on crowd sensing of correlated entities. In: SenSys (2015). https://doi.org/10.1145/2809695.2809715
5. Wang, S., et al.: Scalable social sensing of interdependent phenomena. In: IPSN (2015). https://doi.org/10.1145/2737095.2737114
6. Dong, X.L., et al.: From data fusion to knowledge fusion. In: PVLDB (2014). https://doi.org/10.14778/2732951.2732962
7. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
8. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW (2007). https://doi.org/10.1145/1242572.1242667
9. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: ISWC/ASWC (2007). http://dl.acm.org/citation.cfm?id=1785162.1785216
10. Yin, X., Han, J., Yu, P.S.: Truth discovery with multiple conflicting information providers on the web. In: TKDE (2008). https://doi.org/10.1109/TKDE.2007.190745
11. Zhao, B., Rubinstein, B.I.P., Gemmell, J., Han, J.: A Bayesian approach to discovering truth from conflicting sources for data integration. In: PVLDB (2012). https://doi.org/10.14778/2168651.2168656
12. Tousch, A., Herbin, S., Audibert, J.: Semantic hierarchies for image annotation: a survey. Pattern Recogn. **45**(1), 333–345 (2012). https://doi.org/10.1016/j.patcog.2011.05.017
13. Amit, Y., Fink, M., Srebro, N., Ullman, S.: Uncovering shared structures in multiclass classification. In: ICML (2007). https://doi.org/10.1145/1273496.1273499
14. Beretta, V., Harispe, S., Ranwez, S., Mougenot, I.: How can ontologies give you clue for truth-discovery? An exploratory study. In: WIMS (2016). https://doi.org/10.1145/2912845.2912848

15. Riedel, S., Yao, L., McCallum, A.: Modeling relations and their mentions without labeled text. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010. LNCS (LNAI), vol. 6323, pp. 148–163. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15939-8_10
16. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: ACL (2009)
17. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM J. Comput. **1**(2), 131–137 (1972)
18. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
19. Dong, X.L., Berti-Equille, L., Srivastava, D.: Truth discovery and copying detection in a dynamic world. In: PVLDB (2009). https://doi.org/10.14778/1687627.1687691

Data Warehouses and Recommender Systems

Direct Conversion of Early Information to Multi-dimensional Model

Deepika Prakash(✉)

Department of Computer Engineering, NIIT University, Neemrana 301705, Rajasthan, India
dpka.prakash@gmail.com

Abstract. There are two design methodologies to maintain the single version of the truth in Data Warehouses. Inmon's approach builds a consolidated enterprise DW represented as an ER diagram. Data marts are derived from this and represented as facts and dimensions. Kimball's approach builds a bus of data marts represented as multi-dimensional model that rely on conforming facts and dimensions. However, recent proposals integrate the requirements using the notion of early information. We explore this option by first building a model of early information. The concepts of this model are used in developing an automated conflict resolution mechanism for integration of early information. Finally, we propose an algorithm for the conversion of integrated early requirements into its multi-dimensional form.

Keywords: Early information · Multi-dimension model · Integration
Data marts · Conflicts · Consolidation · Star schema

1 Introduction

At the core of data warehouse design is the notion of the single version of the truth (SVOT). In Inmon's approach [1], the enterprise wide data warehouse (EDW) that captures the SVOT is represented in ER form [2]. Data marts obtained from the EDW are represented in multi-dimensional form. Techniques like [3, 4] perform this conversion. In the approach of Kimball [5], the SVOT is represented in multi-dimensional form and relies on the notion of a bus of data marts. The bus comprises of conformed facts and dimensions across data marts. The pros and cons of the two approaches have been discussed in [2].

A number of techniques exit for obtaining conformed facts and dimensions [6–9]. Recently, a proposal has been made to move integration from the design phase into the requirements phase [10]. This means that instead of conforming facts and dimensions of data marts D1 and D2, the requirements R1 of data mart D1 and R2 of data mart D2 are integrated. The authors of [10] express requirements as 'early information'; the early information has been variously described as [11, 12] fuzzy, first cut, and unstructured. The SVOT is obtained by doing pair-wise integration of requirements.

The integration process [10] starts with identifying the two pieces of early information (EI) to be integrated and finding correspondences between them. Now, if the two pieces are exactly the same, then only one copy of EI is saved. If not, the conflict

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 119–126, 2018.
https://doi.org/10.1007/978-3-319-98812-2_8

resolver is used to resolve conflicts. Finally, the information collator collates the information to give us integrated early information. At the core of the integration process is the conflict resolver. However, the approach to conflict resolution outlined in [10] is semi-automated and requires a fair amount of manual intervention.

Once the requirements have been integrated, the next question is how can we arrive at the facts and dimensions of the DW to-be? [12] follows the approach of Inmon where integrated early information is first converted into ER schema from which facts and dimensions are identified. If Kimball's design is to be adopted then there is a need to convert the integrated early information into multi-dimensional schema directly.

We explore the direct conversion alternative in the paper. The early information model is discussed in the next section of the paper. The integration problem is to take two instantiations and remove conflicts between them. We propose an automated conflict resolution mechanism in Sect. 3. Thereafter, in Sect. 4, we propose an algorithm for the conversion of integrated early requirements into its multi-dimensional form. We conclude the paper in Sect. 5.

2 The Early Information Model

As mentioned above, an instantiation of the early information model yields the requirements of a data mart. That is, the model provides the concepts in terms of which data mart requirements can be expressed.

Figure 1 describes our early information model. As can be seen, information is of three types aggregate, detailed and historical. When information is detailed then it is at its lowest grain. For example, suppose a hotel chain has to purchase beds for all its branches. Detailed information will reflect the beds purchased for *each* branch of the hotel.

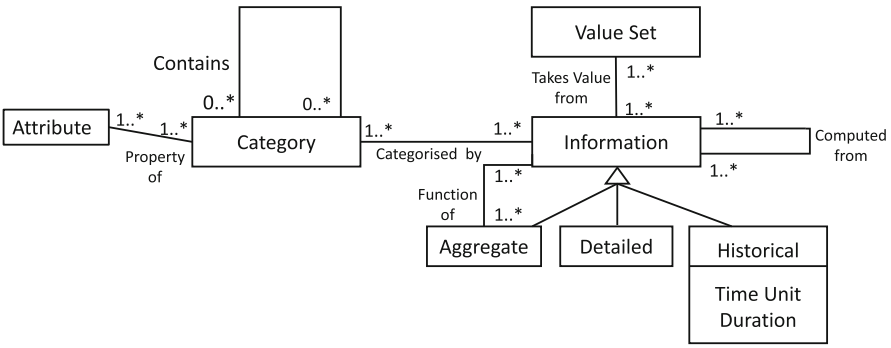


Fig. 1. The early information model

A second type of information is aggregate information. Aggregate information is summarized information. It can be obtained from detailed information or from previously aggregated information. Generally, aggregations come to us as functions like sum, count, min, max etc. This is shown in Fig. 1 by the relationship function-of

between aggregate information and information. An example is *total* number of beds purchased by the hotel group.

There is a third type of information which is Historical and has two properties. Duration tells us the duration for which this information needs to be kept in the DW to-be and Time unit tells us the temporal unit of capturing this information namely daily, monthly, yearly etc.

Information can also be computed from other information (Fig. 1). For example, profit per transaction is information calculated from the difference between sales price and cost price.

Figure 1 also shows that information takes values from a value set. Information can be categorized by category as shown by the information-category relationship in Fig. 1. Let us say that the information is about patient admissions. This can be categorized ward wise and department wise. Figure 1 also shows that categories have attributes associated with them. This tells us the descriptive properties of category.

An inter category relationship, 'contains', has also been defined. For example, in the example considered above, there are two categories, department and ward and a ward is contained in a department or a department 'contains' wards.

The requirements engineer during the requirements engineering task instantiates the information model of Fig. 1. Let us consider the following instantiation. A hospital keeps information about the time spent for consultation which is calculated as the time difference between registration and consultation. Assume that waiting time for all patients visiting a unit and a department is required. Daily information is to be maintained and information for 5 years is to be kept. When we instantiate the model of Fig. 1 with this, we obtain the following:

Information: Time spent for consultation
Computed from: Time of registration; Time of consultation;
Category: Department Contains Unit; Patient
Categorized by :
< waiting time, patient>
< waiting time, unit>
< waiting time, department>
Attributes: (1) Department Attributes: Name
(2) Unit Attributes: Name
(3) Patient Attributes: Registration number, Name, Age, Gender
History Time Unit and Duration: Daily; 5 years

For the rest of this paper we will refer to this information as EI_1.

3 Integration of Early Information

The early information, in accordance with the model of the previous section, is to be structured in multi-dimensional form. However, first early information pieces have to be integrated together. As stated in the Introduction, we concentrate on developing processes and algorithms for automated conflict resolution.

We find the following types of conflicts:

I. ***Name conflicts***: Naming conflicts arise when same information is referred to by different names. Name conflicts can be between categories or between information that is to be integrated. Conflicts can be due to either Homonyms or Synonyms. Synonyms are two or more names referring to the same concept. Schema integration literature [13] suggests many ways of resolving name conflicts. Of these, we have adopted the use of thesaurus for our work.

II. ***Information Type conflicts***: These conflicts arise if in one case information is of detailed type and in another of aggregate type or vice versa.

It is preferred that simple information is stored in the DW to-be. Aggregates can then be created at the level of BI tools. Thus, the type of information in the merged early information will be detailed type.

III. ***Aggregate function conflicts***: It may happen in some cases that both pieces of early information agree that aggregate information is required but they may differ in the type of aggregate function. For example, consider that revenue generated lab test wise is to be kept in the DW to-be. One, say EI_a, may require the average revenue and the other, say EI_b, may require total revenue which indicates the application of function sum.

This conflict can be resolved by applying Algorithm 1 which says that if the aggregate functions are different then integrated early information must be of type detailed. This allows the different aggregate functions to be applied over the detailed information. In other words, we record that the UNION of the functions is to be applied.

Algorithm 1: Integration with aggregate function conflict

Input: early information EI_a and EI_b
Output: integrated EI

```
1:  for Each pair of information, EIa and EIb do
2:    if only one of either EIa or EIb has Function f(x) then
3:      add Function = f(x) to EI
4:      EI:= whichever of EIa or EIb is detailed information
5:    end if
6:    else if EIa has function f1(x) and EIb has function f2(x) then
7:      if f1(x) <> f2(x) then
8:        EI:= Detailed type
9:        Function = f1(x) UNION f2(x)
10:     else keep one copy, say f1(x)
11:   end if
12: end for
```

IV. ***Category conflicts***: Now, we consider Category conflicts. EI_1 is the same as the one illustrated in Sect. 2. Let EI_2 capture information for distribution of personnel in the various departments and units of a hospital. Personnel include patients, doctors, nurses and record clerks. Thus, EI_2 is detailed information categorized by Doctor, Nurse, Patient, Record clerk, Department and Unit.

In other words, while EI_1 categorizes information by 'Department contains Unit', EI_2 categorizes information by 'Department' and 'Unit'. There is no contains relationship for EI_2 between Department and Unit. We refer to these types of conflicts as **category conflicts**.

We propose Algorithm 2 to convert sub category into category.

Algorithm 2*: Sub category to Category conversion*

Input: category, c, contains subcategory, sc
Output: sc as a category

```
1:  for each sc that is in 'contains' c
2:      new_category:= sc
3:      Remove c Contains sc relationship
4:      Add categorized by relationship <Information, new_category >
5:  end for
```

There can also be conflicts between the attributes of the same category. In this case, the attributes of the integrated EI is the UNION of all the attributes of EI_1 and EI_2. In other words, if EI_1 has attributes (A1….An) and EI_2 has attributes (B1….Bm) then EI will have attributes (A1….An, B1….Bm).

V. ***Temporal conflicts***: Here we consider differences in granularity between categories. Information at the lowest grain of the temporal unit needs to be maintained.

4 Deriving Multidimensional Schema from Early Information

In order to progress with DW development, early information based on the information model above has to be converted into a more structured form. We, in this section, propose an algorithm to derive the multidimensional schema from early information. The algorithm is shown below as Algorithm 3.

Algorithm 3: Conversion of early information to multi-dimensional schema

Input: Early Information, I
Output: Snowflake schema for I

```
1:  for Each early information, I do
2:      F:= createfact(I)
3:      for Each computed-from,M, associated with I do
4:          addMeasure(M,I);
5:      for Each category, C associated with I do
6:          D := createDimension(C);
7:          if D does not exist then
8:              for Each attribute, A, linked to category C do
9:                  Add A to D as a Dimensional attribute
10:             end for
11:             Link D to F;
12:         end if
13:         else Link already existing D to F and discard current D
14:         for Each category, cc, contained in C do
15:             SD = createSubDimension(cc)
16:             if SD does not exist then
17:                 for Each attribute, A, linked to cc do
18:                     Add A to SD as a Dimensional attribute
19:                 end for
20:                 Link SD to D
21:             end if
22:             else Link already existing SD to D and discard current SD
23:         end for
24:     end for
25:     if History is to be maintained then
26:         for Each D, linked to F, for which history is required do
27:             Augment(D, timestamp)
28:         end for
29:         Add Dimension Date and link to F
30:     end if
31: end for
```

For each piece of early information, I, the function *createfact* (step 2) takes I as the argument and creates a fact of the multidimensional schema. Each element of the 'computed from' field of the early information, I, becomes measures of the Fact. Steps 3 and 4 show the same.

The *createdimension* function takes each 'category' of I and makes it into a dimension. Internally, this function generates a surrogate key field for the dimension. Attributes of the category become attributes of the dimension. The dimensions are then

connected to the Fact (steps 5–13). Further, if any category has a 'contains' property then createSubDimension creates a sub-dimension between the category and its contained category. The attributes of the sub-dimension are obtained in a manner similar to the one for obtaining attributes for dimensions (steps 14–24).

If history is to be maintained then the fact created is augmented to include a time stamp, TS, by the Augment function (steps 25–31). Further, a Date dimension is added to the schema.

Notice that due to the 'contains' relationship between categories, the resulting multi-dimensional schema is a snowflake schema. Naturally, if there is no contains relationship we obtain a star schema.

After we apply our algorithms of Sect. 3 and Sect. 4, we obtain the full schema shown in Fig. 2.

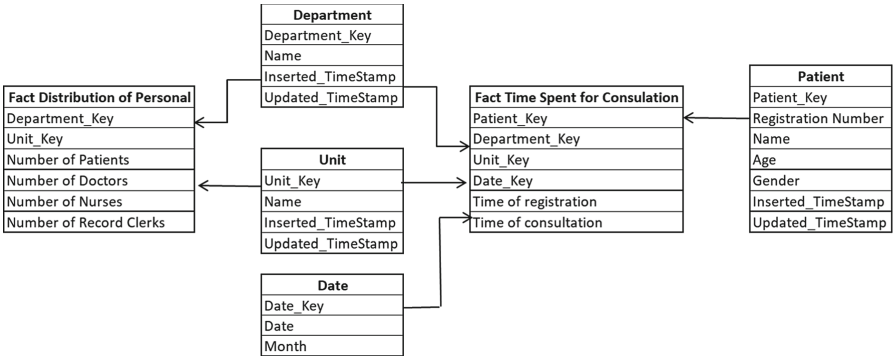


Fig. 2. Multi-dimensional schema obtained after applying Algorithm 3

5 Conclusion

We have taken forward the approach of developing a SVOT by the use of early information. While earlier proposals did elicit requirements as early information, they still relied on converting early information into an intermediate ER diagram. The ER diagram would then have to be converted to multi-dimensional form for the data marts. We have proposed algorithms to convert early information directly into multi-dimensional form. Notice our approach is automated both in the integration stage as well as in the conversion stage.

References

1. Inmon, B.: Building the Data Warehouse, 4th edn. Wiley, New York (2005)
2. Adamson, C.: The Complete Reference Star Schema. McGraw-Hill, New York (2010)

3. Golfarelli, M., Maio, D., Rizzi, S.: Conceptual design of data warehouses from E/R schemes. In: Proceedings of the Thirty-First Hawaii International Conference on System Sciences, vol. 7, pp. 334–343. IEEE, January 1998
4. Moody, L.D., Kortink, M.A.R.: From enterprise models to dimensional models: a methodology for data warehouses and data mart design. In: Proceedings of the International Workshop on Design and Management of Data Warehouses, Stockholm, Sweden, pp. 5.1–5.12 (2000)
5. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. Wiley, Hoboken (2002)
6. Cabibbo, L., Panella, I., Torlone, R., Tre, U.R.: DaWaII: a tool for the integration of autonomous data marts. In: ICDE, p. 158, April 2006
7. Riazati, D., Thom, J.A., Zhang, X.: Inferring aggregation hierarchies for integration of data marts. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6262, pp. 96–110. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15251-1_7
8. Golfarelli, M., Rizzi, S., Turricchia, E.: Modern software engineering methodologies meet data warehouse design: 4WD. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 66–79. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23544-3_6
9. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A., Mayorova, D.: A requirement-driven approach to the design and evolution of data warehouses. Inf. Syst. **44**, 94–119 (2014)
10. Prakash, D., Prakash, N.: A requirements driven approach to data warehouse consolidation. In: 11th IEEE International Conference on Research Challenges in Information Science (RCIS), pp. 449–450 (2017)
11. Prakash, D., Gupta, D.: Eliciting data warehouse contents for policy enforcement rules. Int. J. Inf. Syst. Model. Des. (IJISMD) **5**(2), 41–69 (2014)
12. Prakash, D., Prakash, N.: Towards DW support for formulating policies. In: Ralyté, J., España, S., Pastor, Ó. (eds.) PoEM 2015. LNBIP, vol. 235, pp. 374–388. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25897-3_24
13. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Comput. Surv. (CSUR) **18**(4), 323–364 (1986)

OLAP Queries Context-Aware Recommender System

Elsa Negre[1(✉)], Franck Ravat[2], and Olivier Teste[2]

[1] Paris-Dauphine University, PSL Research Universities, CNRS UMR 7243, LAMSADE, Paris, France
elsa.negre@dauphine.fr

[2] Université de Toulouse, IRIT, CNRS UMR 5505, Toulouse, France
{ravat,teste}@irit.fr

Abstract. It becomes hard and tedious to easily obtain relevant decisional data in large data warehouses. In order to ease user exploration during on-line analytical processing analysis, recommender systems are developed. However some recommendations can be inappropriate (irrelevant queries or non-computable queries). To overcome these mismatches, we propose to integrate contextual data into the recommender system. In this paper, we provide (i) an indicator of obsolescence for OLAP queries and (ii) a context-aware recommender system based on a contextual post-filtering for OLAP queries.

1 Introduction

A substantial effort of the scientific community in the last decades has been to develop data warehousing and on-line analytical processing (OLAP) [1]. Data warehouses (DWs) are built using materialized views [2] based on the multidimensional model, which consists in describing data as a data cube [3]. Contemporary DWs form large multidimensional networks where it becomes hard and tedious to explore and easily obtain relevant decisional data [4].

Context. One possible opportunity is to extend decision support systems with recommender system approaches [5]. The recommendation approaches are popular to provide personalized results into large volumes of data [6]. However, there is few research in the field of OLAP query recommendation [7,8].

Paper Issue. Although these approaches allow recommending queries for a given user, the recommended queries may be not satisfactory because the solution space is limited to queries that existed in the past. The *problem of out of date recommended queries* is the dissatisfaction of the recommender system users. We identified two reasons for the dissatisfaction:

– Irrelevant queries for users, which are not used for long time. These old queries may be considered out of date by users.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 127–137, 2018.
https://doi.org/10.1007/978-3-319-98812-2_9

– Non-computable queries due to data updating: stored data are periodically refreshed, and possibly updated.

Motivating Example. The illustrating example concerns a product company. Users analyze quantities of sales of products according to time and stores. To support this analysis, they query the DW (Fig. 1) where the analysis subject is *Quantity* and the three available analysis axes are *Store*, *Product* and *Time*. Each of them is organized according to a hierarchy of attributes, which represent various granularities. OLAP analysis consists in applying different sessions of queries on the schema defined above. After applying a user query, a recommender system proposes different possible queries that can help the user in his/her analysis.

Given a user query q_c and a set of past queries Ω that the recommender system kept, recommending queries aims to provide a subset $Q_R \subseteq \Omega$ similar to q_c. $Q_R = \{(q_{ri}, \sigma_i) \mid \sigma_i \in [0..1]$ is a similarity value between q_{ri} and $q_c\}$ is an ordered set of recommended queries q_{ri} ranked to the more similar to the less similar regarding to q_c.

Suppose that, today, a user wants to know the quantity of beverages of the range 'Fanta' sold in New York City and San Francisco in 2016 via the query q_c such that: q_c = "Sales of Fanta in San Francisco and New York City in 2016". OLAP queries that can complete the analysis. To suggest additional queries, the OLAP recommender system (RS) calculates similar queries denoted Q_R from the set of stored past queries denoted Ω (previous launched queries between 2005 and 2017). The recommendations returned by the system may be $Q_R = \{(q_{r1}, 0.95), (q_{r2}, 0.94), (q_{r3}, 0.92)\}$ where:

– q_{r1} = "Sales of Cola in New York City in 2012"
– q_{r2} = "Profits related to Fanta in New York City and San Francisco in 2016"
– q_{r3} = "Sales of Fanta in the store M1 of New York City"

Due to periodically refreshing, the DW only contains now products sold in US from 2012 to 2017. In classical RS, the stored queries is kept with non-calculable queries or it is cleaned by deleting out of date queries.

– q_{r1} is out of date because the query handles data related to 2012, which are out of date for the user who focuses on 2016.
– q_{r3} is out of date because the store M1 of New York City is closed since 2011. The M1 data no longer exist in the current DW.

Only $Q'_R = \{(q_{r2}, 0.94)\}$ is usable. Interesting queries, e.g. $Q_R \backslash Q'_R$ are lost because part of data is not kept, and then it is not possible to calculate these out of date queries. Here, q_c is launched in 2017 and q_{r1} could be really interesting for data related to 2016. In the same way, q_{r3} could be interesting for data related to 2016 and either for another store(s), or cities. To overcome these drawbacks, we intend to define an OLAP context-aware RS allowing the recommendation of some past queries such as q_{r1} and q_{r3} that are out of date and/or non calculable into classical RS. We want to extend classic RS where recommended queries are extended with contextual data. In the example, q_{r1} and

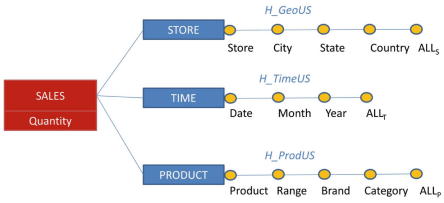


Fig. 1. Example of DW star schema

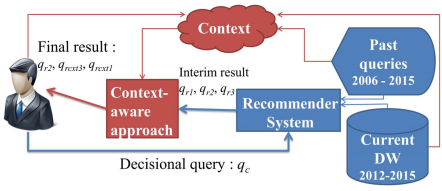


Fig. 2. Our context-aware approach

q_{r3} could be reformulated. The RS may provide a set of contextualized queries $Q_{cxt} = \{(q_{cxt1}, 0.99), (q_{cxt3}, 0.96)\}$ where:

- q_{cxt1} = "Sales of Cola in New York City in 2016".
- q_{cxt3} = "Sales of Fanta in all stores of New York City".

Finally, given a user query q_c, and a past queries set Ω, our approach aims at calculating $Q'_R \cup Q_{cxt} = \{ (q_{cxt1}, 0.99), (q_{cxt3}, 0.96), (q_{r2}, 0.94) \}$, where Q'_R is the set of "usable" recommended queries and Q_{cxt} is a set of out of date queries that are contextualized to be relevant regard to q_c. Figure 2 illustrates the approach to extend the set of recommended queries by reformulating some out of date queries (instead of delete these queries).

Paper Contributions and Organization. In this paper, we propose a context-aware recommender system based on a contextual post-filtering for OLAP queries where we contextualize queries recommended by a classic log-based recommender system. The paper is organized as follows: Sect. 2 presents the related work on OLAP recommender systems and Context-aware recommender systems (CARSs). Section 3 details our indicator of obsolescence and our OLAP queries CARS. Section 4 presents some experiments and results.

2 Related Work

In this section we present some related work about, OLAP recommender systems (RSs) and Context-aware recommender systems (CARSs).

OLAP Recommender Systems. RSs are a particular form of information filtering designed to present information items (e.g., movies, books, ...) that may interest the user. RSs have been studied in many fields, including cognitive science, information retrieval, web and e-commerce. However, some works have focused on recommendations in the field of data warehouses analyzed by OLAP queries and proposed methods and algorithms to assist the user in her/his querying process by proposing relevant queries (items). Among these, some focused on exploiting user profiles and preferences [9], and others focused on the discoveries made during analyses [10] as well as on exploiting logs containing sequences of queries previously run by other users on the same cube [11].

The quality of RSs may be measured by prediction/classification/ranking accuracy, user coverage and recommendation diversity (see [12] for further details). To the best of our knowledge, there does not exist any measure or indicator verifying if the recommendations returned by a RS are out of date.

Context-Aware Recommender Systems. The probably most widely accepted definitions of context is the one of [13] where: "Context is any information that can be used to characterize the situation of an entity". A key accessor to the context in any context-aware system is a well designed model. Some context modeling approaches exist [14,15]. Another complementary approach is to incorporate context. [16] explains that CARSs "generate more relevant recommendations by adapting them to the specific contextual situation of the user". Traditionally, the problem of recommendation can be summarized as the problem of estimating scores for items that have not been seen by a user, i.e. as a prediction problem where the RS predicts the user's ratings for a given item according to a user profile, it is a rating function: $r_{RS} : Users \times Items \rightarrow Ratings$ [16]. With context, the rating function for CARS becomes: $r_{CARS} : Users \times Items \times Contexts \rightarrow Ratings$ [16]. Furthermore, the context can be incorporated in various stages of the recommendation process: (i) pre-filtering, (ii) post-filtering, and (iii) contextual modeling [16]. Finally, [17] concludes that post-filtering seems to be the more efficient.

To the best of our knowledge, there exists no CARS in OLAP field. So, in this paper, our goal is to propose a CARS based on a contextual post-filtering for OLAP queries.

3 OLAP Queries Context-Aware Recommender System

3.1 Definitions

A N-dimensional *Cube* C has for schema $C = \langle D_1, ..., D_N, F \rangle$ where: (i) For $i \in [1, N]$, D_i is a dimension, (ii) F is a fact containing the set of measures.

For a *dimension* D (of the cube C), having for schema an ordered list of m attributes $\{L_j\}$ ($\forall j \in [1, m]$, where m is the depth of the dimension). L_1 is the lowest granularity attribute. Each attribute L_j of the hierarchy is the child of a single parent present at the level of granularity immediately higher L_{j+1} of the hierarchy. In the following, $ADOM(D) = \bigcup_{j=1}^{m} adom(L_j)$, where $adom(L_j)$ is the set of existing values of L_j on C and $ADOM(D)$ is the set of all existing attribute values of dimension D (on C).

Given a cube $C = \langle D_1, ..., D_N, F \rangle$, a *multidimensional query* q_M on C can be represented as $q_M = \langle q_1, ...q_N \rangle$ where $\forall i \in [1, N]$, q_i is a relational query (to obtain the values of the attributes of each dimension D_i). Thus, $q_M = q_1(D_1) \times ... \times q_N(D_N) = \times_{i=1}^{N} q_i(D_i)$.

3.2 ObsoIndic

As displayed in the previous section, there does not exist any indicator verifying if recommended OLAP queries returned by a RS are out of data of use and/or missing data. First of all, we give the definitions of some notations.

– Q_R is the set of all possible recommended queries
– Q'_R is the set of computable recommended queries ($Q'_R \subset Q_R$)

So, we define a new indicator for OLAP queries log-based RS, *ObsoIndic* that measures the number of recommended queries that are obsolete of use (i.e. "old"/irrelevant) or obsolete of data (i.e. non-computable) in relation to the total number of recommended queries such that:

$$ObsoIndic = \frac{|Q_R| - |Q'_R|}{|Q_R|}$$

where $ObsoIndic \in [0; 1]$ and the lower the value, the higher the non-obsolescence of the recommendations.

For example, according to the motivating example (Sect. 1), the log-based RS returns three recommendations (Q_R): q_{r_1}, q_{r_2} and q_{r_3} where q_{r_1} and q_{r_3} are out of date. Our indicator is: $ObsoIndic = \frac{3-1}{3} \approx 0.666$: more than 66% of the recommended queries are irrelevant or non-computable. So this RS needs to enhance its process by integrating our context-aware approach.

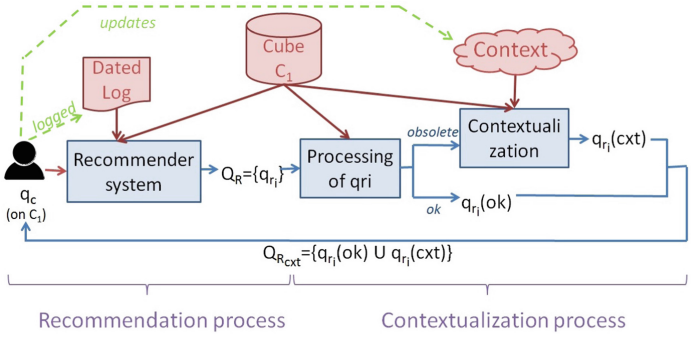


Fig. 3. Our proposition. *Red arrows are inputs and Blue arrows are execution* (Color figure online)

3.3 Global Process

Unlike classic approaches, ours aims at recommending queries that did not exist. Starting from log-based recommended queries, our system exploits additional information to obtain context-aware recommendations (not existing queries). In fact, "old"/irrelevant queries will be updated and non-computable queries will become computable. Our approach mixes decision making process, recommendation process and contextual data to help a decision-maker. We extend a classic RS where returned queries are improved with contextual data. Then, the obtained context-aware recommendations are returned to the decision-maker. Figure 3 displays this additional component. During the recommendation process, the OLAP query, q_c, launched by the decision-maker (i) is logged, (ii) updates the

context and (iii) is the input of the RS. Then, the log-based RS returns a set of recommended queries $Q_R = \{q_{r_i}\}$. Our proposition starts from this point: the contextualization process. Each query q_{r_i} of Q_R is processed. Then:

1. the query q_{r_i} can be processed and is not out of date, in which case, this query is added to the final output $Q_{R_{cxt}}$, or
2. the query q_{r_i} is out of date, in which case, this query does require some updates according to a given context and the new context-aware query q_{cxt_i} is added to the final output $Q_{R_{cxt}}$.

Finally, this set $Q_{R_{cxt}} = \{q_{r_i}\} \cup \{q_{cxt_i}\}$ of context-aware recommendations is returned to the decision-maker.

Specifically, given a current query q_c, and Ω the set of all past queries (query log), the log-based RS proposes a set of recommended queries $Q_R \subseteq \Omega$ that can follow q_c in an analysis session. The set Q_R is composed of queries q_{r_i} ($\forall i \in [1; |\Omega|]$) dated t_i (see Recommendation process part of Fig. 3). Ω can contain relatively old or even obsolete queries. In our approach, to overcome these two cases of obsolescence (because of using only the contents of a log Ω), we want $Q_{R_{cxt}} \subseteq \Delta$ (where Δ represents all the queries that can be launched on the cube C_1 ($\Omega \subset \Delta$)). So when the queries q_{r_i} recommended for q_c are not out of date, we do not act. By cons, when queries q_{r_i} are considered out of date, we want to contextualize them. What we mean by "context-aware queries" is, intuitively, to put the queries "in the style of today". More formally, we have to modify each obsolete query to make them relevant and appropriate to the specific context of the current query q_c (see Contextualization process part of Fig. 3). In more details, during the contextualization process, there are two steps: (i) Processing the q_{r_i} and (ii) Contextualizing the obsolete q_{r_i}.

3.4 Defining/Modeling Context

The context is a set of contextual information [18]. To take fully account of the context of OLAP analysis, we represent it using 5 categories of elements (according to [19]): Time, Individuality, Relations (between users), Activity, Material.

Context modeling is still complicated given the nature of the data and/or contextual information: the model must be able to manage various data sources, their quality and lifetime heterogeneity and their imperfect nature [20]. According to [21], only the ontological model allows a good partial validation of the data and a good formalization of the model. Thus, we use the general ontology of [19] regrouping the five categories.

3.5 Contextualization Process

The contextualization process can be modeled as an algorithm where the inputs are the set Q_R of recommended queries returned by the classic log-based RS (during the recommendation process), the log of queries Ω, the cube on which queries are launched, some contextual data/information and the obsolescence threshold. Concretely, for each recommended query q_{r_i} of Q_R:

- when q_{r_i} is computable and obsolete of use ($isObsoleteU$), the $RecCxtOld$ function updates q_{r_i} into $q_{r_i}(cxt)$ which is added to the final set $Q_{R_{cxt}}$;
- when q_{r_i} is computable and not obsolete of use, q_{r_i} does not require any update and is added to the final set $Q_{R_{cxt}}$;
- when q_{r_i} is not computable, the $RecCxtData$ function updates q_{r_i} into $q_{r_i}(cxt)$ which is added to the final set $Q_{R_{cxt}}$.

Finally, the set $Q_{R_{cxt}}$ of context-aware recommendations is returned by our following Algorithm which complexity is $O(|Q_R| \times t \times k \times |\Omega|^3)$ where $|Q_R|$ is the number of queries in Q_R, t is time to calculate the more complicated query, k is the number of clusters and $|\Omega|$ is the size of the query log Ω.

Algorithm. Context-aware recommended queries: CarsOlap

Require: C_1 the cube on which queries are launched
Q_R : the set of recommended queries returned by the log-based recommender system
$Context$: contextual data/information (ontology)
Ω : the log of queries
$ObsoTh$: the obsolescence threshold
Ensure: $Q_{R_{cxt}}$: the set of context-aware recommended queries
$Q_{R_{cxt}} = \emptyset$
for each q_{r_i} $(\forall i \in |Q_R|)$ **do**
if $q_{r_i} \subset C_1$ (computable on C_1) **then**
if $isObsoleteU(q_{r_i}, \Omega, ObsoTh)$ **then**
Obsolescence of use/irrelevance:
$Q_{R_{cxt}} \leftarrow Q_{R_{cxt}} \cup RecCxtOld(q_{r_i}, Context, \Omega, ObsoTh, C_1)$
else
$Q_{R_{cxt}} \leftarrow Q_{R_{cxt}} \cup q_{r_i}$
end if
else
Obsolescence of data (non-computable):
$Q_{R_{cxt}} \leftarrow Q_{R_{cxt}} \cup RecCxtData(q_{r_i}, Context, C_1)$
end if
end for
Deleting duplications
return $Q_{R_{cxt}}$

The boolean $isObsoleteU(q, \Omega, ObsoTh)$ function returns $true$ when the query q has not been launched in Ω since a certain time ($ObsoTh$ is a threshold, Sect. 4.2 shows how to define its value).

The $RecCxtOld(q, Context, \Omega, ObsoTh, Cube)$ function updates the query q which is obsolete of use with the contextual data $Context$. If the query q contains a selection on some temporal levels, the corresponding values are replaced/upgraded with the TIME category of the $Context$, by keeping the time span between the time the query q was launched and time selections. If the query q does not contains temporal selections, the k-medoid clustering algorithm is used to partition the more recent queries of the log Ω (according to the threshold $ObsoTh$). Each cluster cl_j is represented by a specific query : q^j_{med}. Then, the query q is replaced by the more similar q^j_{med}. Only if the obtained context-aware query is computable on $Cube$ and can be displayed[1] (according to the MATERIAL category of the $Context$), it is returned by the system.

[1] Note that the obtained context-aware query may be update by rolling-up, level by level, until the query can be displayed (according to the MATERIAL category) .

The $RecCxtData(q, Context, Cube)$ function updates the query q which is non-computable on $Cube$, with the contextual data $Context$. If it is a schema problem, i.e. some levels do not exist or are not accessible, and the corresponding dimension exists, then, the query q rolls-up to the 'ALL' level of this dimension. If it is an other schema problem, i.e. some dimensions do not exist or are not accessible, then this dimension is removed from q and some constraints are added through the INDIVIDUALITY category of the $Context$. If it is a data problem, i.e. some values of levels do not exist or are not accessible, and the corresponding level exists, then, all the existing values of the level are displayed and some constraints are adding through the INDIVIDUALITY or TIME categories. Only if the obtained context-aware query is computable on $Cube$ and can be displayed (see footnote 1) (according to the MATERIAL category), it is returned by the system.

4 Experiments

We present the results of the experiments we have conducted to assess the capabilities of our framework. We used synthetic data produced with our own data generator [11]. Both our prototype and our generator are developed in Java using JRE 1.6.0-27 with Postgres 9.1.10 and Mondrian 3.3.0.14703. All tests are conducted with a Core i5-2520M (2.5 Ghz $\times$ 4) with 8 GB of RAM using Linux Ubuntu 12.04.

4.1 Data Set

The process of synthetic log generation is detailed in [11]. Our experiments are conducted with the log-based RS prototype: RecoOLAP [22].

4.2 Results

Obsolescence Threshold. Obsolescence and freshness were studied in the case of OLAP queries [23]. Here, we attempt to experimentally determine the threshold value of obsolescence of a query log. We make 10 successive 10-fold

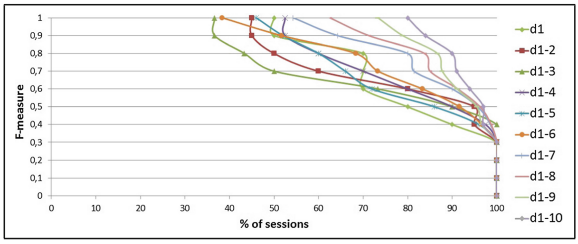


Fig. 4. F-Measure of the recommendations for the 10 validations

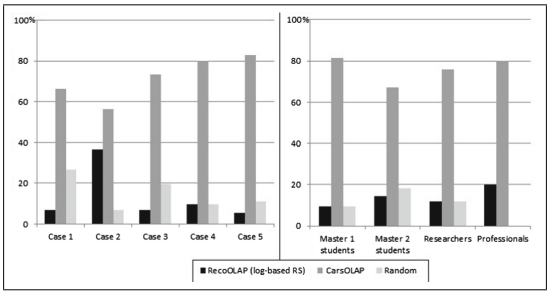


Fig. 5. User feedback analysis (% of pertinent recommendations, left: for each of the 5 cases and right: for different status of responders).

cross validation. First, the generated set of sessions is partitioned, according to the seniority of the sessions in the log, in 10 equally sized subsets, i.e. 10 deciles. We then make a 10-fold cross validation with the 10 deciles. For each such session s_c of size n, we use the sequence of the first $n-1$ queries as the current session, and we compute the recommendations for the n-th query. The generated log contains 610 queries (100 sessions).

Figure 4 shows the inverse cumulative frequency distribution of the recorded F-measure for the 10 validations (d1-10 is the log with 100 sessions, d1-9 contains the 9 more recent deciles, 90 sessions, ... and d1 contains the more recent decile, 10 sessions). This experiment allows us to tune our system in order to choose for the obsolescence threshold the value that achieves best F-measure. First, note that these good results can be explained by the density of the log generated. Second, as many machine learning systems, the more logs are large, the better the quality. So it is normal that $d1-10$ obtains the best results. In the case of $d1$, the log is small but provides a relatively good quality (over 70% of sessions have a F-measure ≥ 0.8). From $d1-2$ to $d1-5$, less than 60% of sessions reach a F-measure ≥ 0.8 (which is not satisfactory). Starting from $d1-6$, results are correct (and close to those of $d1$), we can conclude that we must keep at least the 6 first deciles. Finally, the obsolescence threshold can be defined as $\frac{6}{10}$, e.g. if the log sessions spread over 10 years, sessions/queries over 6 years are obsolete.

User Feedback. User feedback is weakly used to evaluate RSs, due to the difficulty of setting up a protocol and/or the one to involve users (who find the task time-consuming), but a user feedback evaluation allows to position the user at the heart of the evaluation knowing that the vocation of the RS is to satisfy the user. In order to test the quality of the context-aware recommendations, we established a protocol of tests conducted with 72 real users[2] on the FoodMart dataset[3]. Starting from a session of queries (where goal is explained), users are

[2] Students, researchers, professionals from different French universities and enterprises - Master 1 students: 21%, Master 2 st.: 61%, researchers: 15% and professionals: 3%.

[3] http://mondrian.pentaho.org.

offered 3 recommendations (note that users do not know which system returned which recommendation) from (i) the log-based RS (RecoOLAP), (ii) our proposition, *CarsOLAP*, (iii) taken at random from the log. Users should choose which recommendation seems the most relevant for each of the 5 cases[4]. Figure 5 shows that whatever the case and whatever the responder's status, CarsOLAP's recommendations are always the most pertinent (more than 50% even more than 80%). Notice that professionals never chose a recommendation obtained by random.

According to this feedback analysis, it seems that recommendations obtained with our proposition, *CarsOLAP*, a CARS based on a contextual post-filtering, are more relevant than recommendations obtained with a classic log-based RS.

5 Conclusion

In this paper, we exposed the limits of classic log-based RSs when recommending OLAP queries. Indeed, recommended queries can be irrelevant or non-computable. In order to overcome these limitations, we propose (i) an indicator of obsolescence and (ii) to enhance these systems (especially when our indicator value is poor), a process for contextualizing recommended queries and develop a CARS. In OLAP area, we model the concept of Context as an ontology including five categories of elements: Individuality, Activity, Time, Relations and Material. Our contextual post-filtering approach couples a classic recommendation process and a contextualization process where recommended queries returned by classic RS are contextualized to obtain context-aware recommended queries. This allows the system to recommend more relevant and context-aware recommendations.

As future work, we intend to perform experiments on real datasets and scale them up. Some experiments will be perform on more or less obsolete datasets to compare results and, obtain user feedbacks to compare the recommendation quality. Our CARS should be applicable both to detailed as well as aggregated data. Our system must be able to choose whether the query runs or not on detailed data or on aggregates. In the same way, our system should remove irrelevant aggregates and/or create new ones. Finally, other approaches can be proposed to modeling and integrating context into CARSs, taking into account for example, the multidimensional nature of context. Furthermore, we hope that RSs will become more than context-aware RSs and perhaps even context-driven RSs.

References

1. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. **26**(1), 65–74 (1997)
2. Hammer, J., Garcia-Molina, H., Widom, J., Labio, W., Zhuge, Y.: The stanford data warehousing project. IEEE Data Eng. Bull. **18**(2), 41–48 (1995)

[4] The questionnaire (in French) is available online: http://www.lamsade.dauphine.fr/~negre/questionnaire.html.

3. Gray, J., et al.: Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. DMKD **1**(1), 29–53 (1997)
4. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and OLAP multidimensional networks. In: SIGMOD 2011, New York. ACM, pp. 853–864 (2011)
5. Bobadilla, J., Ortega, F., Hernando, A., GutiéRrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
6. Koutrika, G., Bercovitz, B., Garcia-Molina, H.: Flexrecs: expressing and combining flexible recommendations. In: SIGMOD 2009, New York, pp. 745–758. ACM (2009)
7. Bimonte, S., Negre, E.: Evaluation of user satisfaction with OLAP recommender systems: an application to recoolap on a agricultural energetic consumption datawarehouse. IJBIS **21**(1), 117–136 (2016)
8. Ravat, F., Teste, O.: Personalization and OLAP databases. In: New Trends in Data Warehousing and Data Analysis, pp. 1–22 (2009)
9. Golfarelli, M., Rizzi, S., Biondi, P.: myOLAP: an approach to express and evaluate OLAP preferences. IEEE Trans. Knowl. Data Eng. **23**(7), 1050–1064 (2011)
10. Sarawagi, S.: User-adaptive exploration of multidimensional data. In: VLDB, pp. 307–316(2000)
11. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query recommendations for OLAP discovery-driven analysis. IJDWM **7**(2), 1–25 (2011)
12. Negre, E.: Information and Recommender Systems. Advances in Information Systems Set. WILEY-ISTE (2015)
13. Dey, A.K.: Understanding and using context. Pers. Ubiquitous Comput. **5**(1), 4–7 (2001)
14. Strang, T., Popien, C.L.: A context modeling survey. In: UbiComp, Nottingham, pp. 31–41, September 2004
15. Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F.A., Tanca, L.: A data-oriented survey of context models. SIGMOD Rec. **36**(4), 19–26 (2007)
16. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 217–253. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_7
17. Panniello, U., Gorgoglione, M.: Context-aware recommender systems: a comparison of three approaches. In: CEUR Workshop Proceedings, vol. 771 (2011)
18. Zimmermann, A., Lorenz, A., Oppermann, R.: An operational definition of context. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 558–571. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74255-5_42
19. Negre, E.: Pertinent context information for OLAP applications. In: KMIKS 2017 (2017)
20. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: models and approach. Pervasive Mob. Comput. **2**(1), 37–64 (2006)
21. Soualah Alila, F.: CAMLearn: a context-aware mobile learning recommender system. Application to M-Learning Domain. Ph.D. thesis, Dijon, France (2015)
22. Giacometti, A., Marcel, P., Negre, E.: Recommending multidimensional queries. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 453–466. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03730-6_36
23. Röhm, U., Böhm, K., Schek, H.J., Schuldt, H.: Fas: A freshness-sensitive coordination middleware for a cluster of olap components. In: VLDB, pp. 754–765 (2002)

Combining Web and Enterprise Data for Lightweight Data Mart Construction

Suzanne McCarthy(✉), Andrew McCarren(✉), and Mark Roantree(✉)

Insight Centre for Data Analytics, School of Computing,
Dublin City University, Dublin 9, Ireland
suzanne.mccarthy@insight-centre.org,
{andrew.mccarren,mark.roantree}@dcu.ie

Abstract. The Agri sector has shown an exponential growth in both the requirement for and the production and availability of data. In parallel with this growth, Agri organisations often have a need to integrate their in-house data with international, web-based datasets. Generally, data is freely available from official government sources but there is very little unity between sources, often leading to significant manual overhead in the development of data integration systems and the preparation of reports. While this has led to an increased use of data warehousing technology in the Agri sector, the issues of cost in terms of both time to access data and the financial costs of generating the Extract-Transform-Load layers remain high. In this work, we examine more lightweight data marts in an infrastructure which can support on-demand queries. We focus on the construction of data marts which combine both enterprise and web data, and present an evaluation which verifies the transformation process from source to data mart.

1 Introduction

Agri companies are increasingly making use of data analytics and data warehouse technologies. Start-ups and research groups are emerging for the purpose of addressing the need for data analytics unique to the Agri sector [1,7]. Extract-Transform-Load (ETL) as a framework for data integration gained popularity in the 1970's and quickly became the standard process for data integration. Traditional ETL grabs a 'snapshot' of the source data at given intervals, be it daily, weekly etc. and updates the data warehouse or data marts [14]. This happens pro-actively at these intervals and all the source data is extracted, transformed according to relevant rules and loaded into a warehouse where it remains until it is needed for user queries. The primary issue with this approach is that it is very costly in terms of both time and resources. As the integration effort is generally significant, it is not easy to modify the ETL system, meaning that the addition of new sources is a major task.

Research funded by Science Foundation Ireland under grant number SFI/12/RC/2289.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 138–146, 2018.
https://doi.org/10.1007/978-3-319-98812-2_10

Motivation. The Kepak Group [6] is a leading Irish meat producer and our industry partner and provided the business requirements that drive our case study for this project. Using one of these requirements, we have specified a data mart to be built and populated using our ETL framework. It is often the case that only a small subset of all the data loaded is actually required for a user's query. However, the user must still wait for the loading of the data update in its entirety [5], a burden for data scientists who require near real-time data to make predictions. Traditional ETL is typically run on a batch basis, for example on a weekly or daily schedule. All of the data from the input data source(s) are transformed and loaded to the target warehouse, with no regard for how, when or if it will be used at query-time. This is less than ideal for two main reasons: it is heavy on resources and creates redundancy; this time-consuming process also means that the warehouse is constantly out of date. In [5], researchers focus on shortening the update interval using Active Warehousing. However, doing so increases the burden on the transactional and analytical databases, as it requires performing more frequent extractions, scheduled concurrently with the frequent loads, while still not addressing the issue of redundancy in the data.

Contribution. In this paper, we present a new framework which presents a more lightweight approach to the ETL process. The framework also provides a methodology to integrate unknown data sources, from both enterprise and web environments. With the aid of an (Agri) ontology and transformation templates, the Transform and Load components are used to populate the data marts directly. As the process is largely automatic, it has a significant benefit to end users who can import new data into data marts for analysis and predictions. However, this fast transformation of unknown sources may introduce errors and thus, our evaluation contains validation routines to verify transformed data.

Paper Structure. The remainder of this paper is structured as follows: In Sect. 2, we present a review of related research in this area; Using a real world case study from a test application developed with our industry partner, Sect. 3 shows our approach to importing new data sources; Sect. 4 continues the case study with a description of data mart construction and population; in Sect. 5, we present our evaluation which validates our transformation process; and finally, in Sect. 6, we present our conclusions.

2 Related Research

As the primary focus of this paper is to build an automated ETL process to combine web and enterprise data into data marts, we focus on similar work in this area. While data integration is now a well-understood problem and ETL architectures comprise mature processes, research into systems that attempt a form of auto-integration between enterprise and web sources, or web sources alone, remains topical.

In [10], the difficulty in automating a process to assign the mappings and transformations required for ETL is addressed. While information about the

semantics of the data, the necessary constraints and requirements may be hard to find or unreliable, the design of an ETL process hinges on this information. Therefore, the authors propose making use of an ontology in order to provide this domain-specific information to drive the transformation stage of the ETL process. Our approach also requires a domain-specific ontology to drive the transformation stage of the ETL process but we propose a number of metadata structures drawn from the ontology for a more lightweight ETL process which can support on-demand querying.

In [13], the authors present an ontology-based agricultural knowledge fusion method based on recent advances in the area of data fusion such as the semantic web. Their ontology defines an integrated hierarchy of agricultural knowledge: definitions and relationships. The purpose was to use an ontology to drive a knowledge fusion method to resolve disparity when integrating Agri data sources to create knowledge as well as analyse the data. We also make use of an ontology but instead combine it with transformation templates to resolve the heterogeneities between enterprise and web sources.

In [2], researchers focus on reducing latency in the data warehouse as a decision-making tool. Similar to our work, they regard the creation of a real-time data warehouse as a continuous data integration environment and end-to-end automation of the ETL process. They propose an architecture that facilitates a more streamlined approach to ETL by moving the data between the different layers of the architecture without any intermediate file storage. However, data is loaded as it becomes available and not as it is needed by the user. Our approach populates data marts from the Data Lake as they are required by the end user.

3 Data Importation Process

In traditional ETL systems, warehouse updates occur on a periodic basis through the ETL process and normally represents a significant overhead. The various components of Extract, Transform and Load are separated in our framework where the Extract process imports to the Data Lake only and involves the use of a Metabase as a management tool for the Data Lake.

3.1 Data Lake Storage

A Data Lake is a repository for large amounts of data from multiple sources, wherein the data is stored in its native format. This makes it a very lightweight and low-overhead data storage area. In our system, we adapt the concept slightly and perform a very simple transformation on each of the imported datasets, which are then stored as sets of attribute-value (A-V) pairs.

When a new data source is imported, metadata is recorded as an *Import Template*, and updated again when the data is transformed, queried and/or loaded. The Import Template then can provide the system with basic details (metadata) of the source data. An example of the Import Template can be found at [9].

3.2 Transformation

A transformation template (*Transformer*) is created for each individual data source and it stores each of the native terms used in the data and maps each of them to a standard term. In this research, we are using the Agri warehouse model described in [8]. All data marts must conform with the common model: they are cubes derived from the base schema. The base schema is a star schema comprised of 24 dimensions and 26 fact tables.

The output of the following 5-step process is a dataset which is *ready to load* into the data mart. Given a set $S = \{(A_1, V_{11}), (A_1, V_{12}), (A_1, _{V13})...(A_4, V_{43})\}$

1. Retrieve attribute-value pair.
2. Map attribute to standard term found in Transformer.
3. Map value to standard term found in Transformer.
4. Create transformed attribute-value pair.
5. Update Import Template to indicate transformation status.

3.3 Load

The Load process initiates a MySQL Insert statement and populates parameters before execution. Both the Import Templates and the Transformers are used again in the Loading process. To fulfil the Load process, the set of transformed attribute-value pairs are first split into batches, where each batch contains a *single item* of *each* measure and its associated dimension data. These are then used to populate a MySQL command to insert data to the data mart. The Import Template informs the process of the size of each batch and the target mart. Further details of the Transformation and Loading workflows can be found at [9].

4 Data Mart Construction

4.1 Overview

Kepak, our Agri business partner, have a requirement to analyse trade prices of beef commodities and wish to do so using their own enterprise data, combined with selected web based information. The 5 data sources are described below and, in the following section, we provide details for the Eurostat import and transformation process.

- The Eurostat [4] website is the source for EU trade data which is publicly available.
- The United States Department of Agriculture (USDA) [12] publishes Agri trade data figures which can be downloaded in bulk.
- StatCan [11] is the Canadian National Statistics agency and publishes economic, social and census data.
- Comtrade [3] is the U.N. international trade statistics database.

– Kepak Group [6] have exported from their own operational databases.

The above sources provide dimensional data to the following source attributes:

– **reporter**: the country that is reporting this data
– **partner**: the country with whom the reporter is trading
– **product**: the commodity being traded
– **flow**: the direction of the trade, i.e. imports or exports.

All sources additionally have a time attribute but the **date** dimension in the warehouse is static and pre-populated with a fixed range of dates before loading time. Therefore, it is not part of the evaluation process presented in Sect. 5. All web sources use **trade weight**, measured in either tons or kg, and **trade value**, in their local currency, as their measures. The enterprise data uses both of these along with two additional measures: **offcut_value** and **yield**.

Beef trade data is extracted from each of these sources. This data will then populate a beef trade data mart with the following dimensions: **dim_geo**, **dim_trade_product**, **dim_trade_flow**, **dim_date_monthly**; and measures: **trade_weight**, **trade_value**, **offcut_value** and **yield**. Both the reporter and partner source attributes will populate the **dim_geo** dimension.

4.2 Details for Eurostat Import

The input for this process (Fig. 1) is a file download from the website. Example 1 shows a single row of the Eurostat data in the attribute-value pairs which is the format for the Data Lake.

DEC_LAB	PARTNER_LAB	PRODUCT	PRODUCT_LAB	FLOW_LAB	PERIOD	INDICATORS	INDIC_VALUE
France	Netherlands	2011000	CARCASES OR HALF-CARCASES OF BOVINE...	EXPORT	201708	VALUE_1000EURO	828.68
France	Netherlands	2011000	CARCASES OR HALF-CARCASES OF BOVINE...	EXPORT	201708	QUANTITY_TON	198.6
France	Netherlands	2011000	CARCASES OR HALF-CARCASES OF BOVINE...	EXPORT	201709	VALUE_1000EURO	773.63
France	Netherlands	2011000	CARCASES OR HALF-CARCASES OF BOVINE...	EXPORT	201709	QUANTITY_TON	184.1

Fig. 1. Eurostat web output

Example 1. Eurostat data in Data Lake
S={(PERIOD, 201708),
(DECLARANT_LAB, France),
(PARTNER_LAB, Netherlands),
(FLOW_LAB, EXPORT),
(PRODUCT, 2011000),
(PRODUCT_LAB,CARCASES OR HALF-CARCASES OF BOVINE ANIMALS, FRESH OR CHILLED),
(INDICATORS, QUANTITY_TON),
(INDICATOR_VALUE, 198.6) }

4.3 Eurostat Transformation Process

The input for this process is the Eurostat data in its Data Lake format, as seen in Example 1. For each attribute and value, the Eurostat Transformer supplies the standard term which should replace it (though it is not always the case that the source's original term is different from the standard term). The output of this process is seen in Example 2.

Example 2. Eurostat data after Transformation
S'={ (yearmonth, 201708),
(reporter, FRANCE),
(partner, NETHERLANDS),
(flow, exports),
(product_code, 2011000),
(product_desc,CARCASES OR HALF-CARCASES OF BOVINE ANIMALS, FRESH OR CHILLED),
(unit, ton),
(value, 198.6) }

5 Evaluation

As with any automated transformation process, validation of the process itself is crucial. The goal of this paper is on the overall framework which moves data from source to data mart and the focus of the evaluation to validate transformations. This involves two sets of experiments: the first validates that all *measure* data was loaded correctly in its entirety from source to (warehouse model) data mart; the second validates that *dimensional* data underwent valid transformation. Possible states that may result during the Loading of both fact and dimension tables are:

- **Attribute error**: A mismatch between a data point from the source file and the data mart because dimensions were extracted from the Data Lake in the wrong order. This could occur with both dimensional and measure data.
- **Value error**: Missing values where a value is accidentally skipped during the Loading process means that the datasets have different cardinalities. This could occur with both dimensional and measure data.
- **Transformer error**: Errors during creation of the Transformer means that the data is incorrectly NULL. This can only occur for dimensional data.
- **True**: The validation test passes.

Measure Validation: Ordered List Test. The purpose of the Ordered List (OL) Test is to ensure that the correct number of data points were loaded to the mart (no missing or duplicated data) and that values that should remain unchanged, did so. The OL test runs against every measure in each of the data sources. The inputs for this process are an ordered list of measure values obtained from the Fact table, and an ordered list of measure values extracted from the

source file. It tests for Attribute errors and Value errors, and returns True if none are found.

Frequency Pattern (FP) Test for Dimension Values. Dimensional data is treated as non-numeric and values may be required to change during the Transformation process. The purpose of this test is to ensure that term transformations are *consistent*. This test is run on every populated dimension. Two sorted lists are constructed: the frequency count of all distinct dimensional values found within the source file, and the distinct frequency count of all foreign keys found within the fact table, both sorted in ascending order. If the frequencies of the values are different, then either two source original terms have been mapped to the same standard term or a single original term has been mapped to more than one standard term, i.e. term mappings have not been consistent.

The two sorted lists are compared and tested for all of the potential errors - attribute, value and transformer, and returns True if none are found.

5.1 Results Overview

The results of all validation tests run gave a result of 6 failures out of 28 tests giving a success rate of 78.57%. On investigating the causes for the failures, all the errors listed previously were found.

- **Attribute error**: For example with USDA, the **dim_trade_product** foreign key was assigned to the **dim_trade_flow** value. This shows the importance of importing the dimension values in the order in which they are expected to populate the dimensions.
- **Value error**: For example with Comtrade trade value, a data point was skipped as a result of an error during Loading.
- **Transformer error**: For the dimensional data of our business partner, there is a high number of dimensions and not all were imported as the Agri model did not capture this level of dimensionality.

5.2 Detailed Eurostat Results

The results for the validation of the Loading process for a Eurostat dimension and a measure are given.

Figure 2 shows the Frequency Pattern of the **product** source attribute as extracted from the **dim_trade_product** dimension for Eurostat. The label **trade_product_sk** refers to the ID of the fact's foreign key link with the **dim_trade_product** dimension. It can be seen that the terms in the source file and the transformed data in the data mart have the same frequency pattern and that the source original terms were mapped correctly to a standard term.

For the sake of brevity, the entire results of the Ordered List tests on the measure will not be displayed. Instead we have summed the values of **trade_weight** from the source file (TWS) and the values of **trade_weight** from the fact table (TWF).

$$\sum(tws \in TWS) - \sum(twf \in TWF) = 0$$

Frequency data from source file		Frequency data from data mart		
PRODUCT_LAB	freq.	trade_product_sk	product_desc	freq.
FRESH OR CHILLED BOVINE MEAT, BONELESS	1402	183	FRESH OR CHILLED BOVINE MEAT, BONELESS	1402
FRESH OR CHILLED BOVINE CUTS, WITH BONE IN (EXCL. CARCASES AND HALF-CARCASES, "COMPENSATED QUARTERS", FOREQUARTERS AND HINDQUARTERS)	944	179	FRESH OR CHILLED BOVINE CUTS, WITH BONE IN (EXCL. CARCASES AND HALF-CARCASES, "COMPENSATED QUARTERS", FOREQUARTERS AND HINDQUARTERS)	944
CARCASES OR HALF-CARCASES OF BOVINE ANIMALS, FRESH OR CHILLED	622	168	CARCASES OR HALF-CARCASES OF BOVINE ANIMALS, FRESH OR CHILLED	622
UNSEPARATED OR SEPARATED HINDQUARTERS OF BOVINE ANIMALS, WITH BONE IN, FRESH OR CHILLED	528	174	UNSEPARATED OR SEPARATED HINDQUARTERS OF BOVINE ANIMALS, WITH BONE IN, FRESH OR CHILLED	528
UNSEPARATED OR SEPARATED FOREQUARTERS OF BOVINE ANIMALS, WITH BONE IN, FRESH OR CHILLED	504	173	UNSEPARATED OR SEPARATED FOREQUARTERS OF BOVINE ANIMALS, WITH BONE IN, FRESH OR CHILLED	504
"COMPENSATED" QUARTERS OF BOVINE ANIMALS WITH BONE IN, FRESH OR CHILLED	384	169	"COMPENSATED" QUARTERS OF BOVINE ANIMALS WITH BONE IN, FRESH OR CHILLED	384

Fig. 2. Eurostat product dimension Frequency Pattern

The results of the Eurostat case study data shows that the data retains its integrity throughout the ETL process and the original data could, if necessary, be re-created from the data mart by reversing the transforms by the same process. This supports our vision of a lightweight ETL process that makes use of metadata structures, as opposed to a high-overhead traditional data integration process.

6 Conclusions

Data warehouses provide the basis for powerful analytical tools but are expensive to build and support, and tend to be very slow to incorporate new data sources. In this work, we looked at creating more lightweight data marts in order to accommodate new data sources that are selected by end users. We presented a framework which uses a common Agri model to which data marts must conform, and a method to transform data sources into the conforming data mart. A real world case study was specified by our Agri business partner which used their own enterprise data together with 4 selected web sources. Our evaluation used a number of validation techniques to ensure that data extracted from source and loaded into data marts were accurate. An area of work not addressed in this research is the automatic construction of the Transformer which is part of current research. Our overall aim with this platform is to provide Query on Demand data marts which extract from the data lake as required by the user.

References

1. DIT Agriculture Analytics Research Group (2018). http://www.agrianalytics.org/
2. Bruckner, R.M., List, B., Schiefer, J.: Striving towards near real-time data integration for data warehouses. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2002. LNCS, vol. 2454, pp. 317–326. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46145-0_31
3. UN Comtrade (2018). https://comtrade.un.org//
4. Eurostat: Your key to European statistics (2018). http://ec.europa.eu/eurostat/about/overview
5. Kargın, Y., Pirk, H., Ivanova, M., Manegold, S., Kersten, M.: Instant-on scientific data warehouses. In: Castellanos, M., Dayal, U., Rundensteiner, E.A. (eds.) BIRTE 2012. LNBIP, vol. 154, pp. 60–75. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39872-8_5
6. Kepak Group (2018). https://www.kepak.com/
7. MCR Agri Analytics (2018). http://www.mcragrianalytics.com/
8. McCarren, A., McCarthy, S., Sullivan, C.O., Roantree, M.: Anomaly detection in agri warehouse construction. In: Proceedings of the ACSW, pp. 1–17. ACM Press (2017)
9. McCarthy, S., McCarren, A., Roantree, M.: An Architecture and Services for Constructing Data Marts from Online Data Sources. Insight Report 2018–1, April 2018. http://doras.dcu.ie/22386/1/DEXA-TechnicalReport-2018.pdf
10. Skoutas, D., Simitsis, A., Sellis, T.: Ontology-driven conceptual design of ETL processes using graph transformations. J. Data Sem. **13**, 120–146 (2009)
11. Statistics Canada (2018). https://www.statcan.gc.ca/eng/start
12. United States Department of Agriculture (2018). http://www.ers.usda.gov/data-products/chart-gallery/detail.aspx?chartId=40037
13. Xie, N., Wang, W., Ma, B., Zhang, X., Sun, W., Guo, F.: Research on an agricultural knowledge fusion method for big data. In: Data Science Journal (2015)
14. Zhu, Y., An, L., Liu, S.: Data updating and query in real-time data warehouse system. In: 2008 International Conference on Computer Science and Software Engineering, CSSE 2008, Wuhan, China, pp. 1295–1297 (2008)

FairGRecs: Fair Group Recommendations by Exploiting Personal Health Information

Maria Stratigi[1], Haridimos Kondylakis[2], and Kostas Stefanidis[1(✉)]

[1] University of Tampere, Tampere, Finland
{maria.stratigi,kostas.stefanidis}@uta.fi
[2] ICS-FORTH, Heraklion, Greece
kondylak@ics.forth.gr

Abstract. FairGRecs aims to offer valuable information to users, in the form of suggestions, via their caregivers, and improve as such the opportunities that users have to inform themselves online about health problems and possible treatments. Specifically, FairGRecs introduces a model for group recommendations, incorporating the notion of fairness. For computing similarities between users, we define a novel measure that is based on the semantic distance between users' health problems. Our special focus is on providing valuable suggestions to a caregiver who is responsible for a group of users. We interpret valuable suggestions as ones that are both highly related and fair to the users of the group.

1 Introduction

During the last decade, the number of users who look for health and medical information has dramatically increased. However, it is still very hard for a patient to accurately judge the relevance of some information to his own case and to identify the quality of the provided information. The optimal solution for patients, however, is to be guided by healthcare providers to more optimal resources over the Web [1]. Delivering accurate sources to a patient, increases his/her knowledge and changes the way of thinking which is usually referred as patient empowerment. As a result, the patient's dependency for information from the doctor is reduced. Moreover, patients feel autonomous and more confident about the management of their disease [9]. To achieve this, health providers have the history of their patient's and their interests, in order to make an informed decision about the information that would likely be beneficial for the patients. However, health providers have less and less time to devote to their patients. As such, guiding each individual patient appropriately is a really difficult task. On the other hand, the use of group-dynamics-based principles of behavior change, have been shown to be highly effective in enhancing social support. In those cases, a caregiver guides patient groups to more optimal resources over the Web. However, if identifying online information content for a single patient is a difficult task, identifying information for a group of participants is a really challenging one.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 147–155, 2018.
https://doi.org/10.1007/978-3-319-98812-2_11

To this direction, we focus on recommending interesting health documents selected by health professionals, to groups of users, incorporating the notion of fairness, using a collaborative filtering approach. Our motivation is to offer a list of recommendations to a caregiver who is responsible for a group of patients. The recommended documents need to be relevant, based on the patients current profiles. To exploit patients profiles, we use the data stored in their personal health-care record (PHR) data. These patients do not necessarily suffer from the same health problems, but a variety of them. As such, we introduce the notion of fairness in the recommendation process.

More specifically, the contributions of our work are the following: (a) We demonstrate the first group recommendation model incorporating fairness in the health domain; (b) We propose a novel semantic similarity function that takes into account the patients medical profiles, showing its superiority over a traditional measure; (c) We introduce a new aggregation method that encapsulates the notion of fairness; (d) We explore 5 different aggregation methods; (e) We present the first synthetic dataset and the corresponding engine for constructing it, for benchmarking works in the area. To our knowledge, this is the first work that introduces fair group recommendations in the health domain. A preliminary abridged version of this paper appears in [8].

2 Single User Recommendations

Assume a recommender system in the health domain, where I is a set of data items to be rated and U is the set of patients in the system. A patient, or user, $u \in U$ might rate an item $i \in I$ with a score $r(u, i)$, as in $[1, 5]$. Typically, the cardinality of the item set I is high and users rate only a few items. The subset of users that rated an item $i \in I$ is denoted by $U(i)$, while the subset of items rated by a user $u \in U$ is denoted by $I(u)$.

For the items unrated by the users, recommender systems estimate a relevance score, denoted as $relevance(u, i)$, $u \in U$, $i \in I$. To estimate the relevance score of an item, we follow the collaborative filtering approach. First, similar users are located via a *similarity function* that evaluates the proximity between two users. Then, items relevance scores are computed for individual users, taking into account their most similar users. Instead of only using classical similarity notions, we exploit the similarity in patient profiles (their diseases), improving the quality of the recommendations.

Similarity Based on Ratings. Two users are similar if they have rated data items in a similar way, i.e., they share the same interests. We calculate their similarity based on their ratings, by exploiting the Pearson correlation metric:

$$RatS(u, u') = \frac{\sum_{i \in X}(r(u,i) - \mu_u)(r(u',i) - \mu_{u'})}{\sqrt{\sum_{i \in X}(r(u,i) - \mu_u)^2}\sqrt{\sum_{i \in X}(r(u',i) - \mu_{u'})^2}}, \text{ where } X = I(u) \cap I(u'),$$

μ_u is the mean of the ratings in $I(u)$.

Similarity Based on Semantic Information. In the health domain, usually two people have similar interest in health documents if they have similar health problems. ICD10[1] is a standard medical classification ontology, which we exploit to record and identify similarities between health problems and eventually between users. The ICD10 taxonomy can be represented as a tree, with health problems as its nodes. In the 2017 version of ICD10, there are 4 levels in the tree, in addition to the root level. Sibling nodes that belong to lower levels share greater similarity than siblings that belong to upper levels. Because of this, we assign different weights to nodes according to their level. These weights will help us differentiate between siblings nodes in the various levels; we want sibling nodes in the higher levels to share greater similarity than those in the lowest. Formally, for a node A in the ontology tree, $weight(A) = w * 2^{maxLevel - level(A)}$, where w is a constant, $maxLevel$ is the maximum level of the tree and $level(A)$ is a function that returns the level of each node. In addition, let $anc(A)$ be the direct ancestor of A, and LCA(A,B) be the lowest common ancestor of the nodes A and B. For computing the distance between A and B, we compute their distance from $LCA(A, B) = \text{C}$. The distance between A and C is calculated by accumulating the weight of each node in the path, as $dist(A, C) = \sum_{n \in path(A,C)} weight(n)$. In overall, the similarity between A and B is: $simN(A, B) = 1 - \frac{dist(A,C)+dist(B,C)}{maxPath*2}$, where $maxPath = dist(root, L)$. L is a leaf node in the highest level.

Overall Similarity Between Two Users. Let $Problems(u)$ be the list of health problems of user $u \in U$. As such, given two users u and u', we calculate their overall similarity by taking into consideration all possible pairs of health problems between them. Specifically, we take one by one all the problems in $Problems(u)$ and calculate the similarity with all the problems in $Problems(u')$. For each distinct problem from u, we take into account only the health problem of u' that has the maximum similarity.

Definition 1 (SemS). *Let u and u' be two users in U. The similarity based on semantic information between u and u' is defined as:* $SemS(u, u') = \frac{\sum_{i \epsilon Problems(u)} ps(i,u')}{|Problems(u)|}$, *where* $ps(i, u') = max(\forall_{j \epsilon Problems(u')}\{simN(i, j)\})$.

Single User Rating Model. Let P_u denote the set of the most similar users to u, hereafter, referred to as the *peers* of u. If u has expressed no preference for an item i, the relevance of i for u is estimated as: $relevance(u, i) = \frac{\sum_{u' \in (P_u \cap U(i))} S(u,u')r(u',i)}{\sum_{u' \in (P_u \cap U(i))} S(u,u')}$, where S is either $RatS$ or $SemS$.

After estimating the relevance scores of all unrated user items for a user u, the items A_u with the top-k relevance scores are suggested to u.

3 Group Recommendations

Our goal is to provide valuable suggestions to a caregiver who is responsible for a group of patients. We interpret valuable suggestions as suggestions that are both highly related and fair to the patients of the group.

[1] http://www.icd10data.com/.

Group Rating Model. Most previous works focus on recommending items to individual users. Recently, group recommendations that make recommendations to groups of users instead of single users (e.g., [5,6]), have received considerable attention. Commonly, a method for computing group recommendations first estimates the relevance scores of the unrated items for each user in the group, and then, aggregates these predictions to compute the suggestions for the group. Formally, the relevance of an item for a group is computed as follows:

Definition 2 (Relevance). *Let U be a set of users and I be a set of items. Given a group of users G, $G \subseteq U$, the group relevance of an item $i \in I$ for G, such that, $\forall u \in G$, $\nexists rating(u,i)$, is: $relevanceG(G,i) = Aggr_{u \in G}(relevance(u,i))$.*

As in single user recommendations, the items with the top-k relevance scores for the group are recommended to the group.

Fairness in Group Recommendations. Given a particular set of recommendations for a caregiver, it is possible to have a user u that is the least satisfied user in the group for all items in the recommendations list, that is, all items are not related to u. Therefore, although the caregiver may like as a whole the set of recommendations, the package selection is not fair to u. In actual life, where the caregiver is concerned for the needs of all patients in his group, we should recommend items that are both strongly relevant and fair to the majority of the group members. In particular, to increase the quality of the recommendations for the caregiver, we consider, similar to [7], a fairness measure that evaluates the goodness of the recommendations as a set. This way, given a user u and a set of recommendations D, we define the degree of fairness of D for u as $fairness(u,D) = \frac{|X|}{|D|}$, where $X = A_u \cap D$.

Intuitively, the fact that the group recommendations contain some highly relevant items to u, makes both u and his caregiver tolerant to the existence of other items that are not highly related to u, considering that there are other members in the group who may be related to these items. Then, the fairness of a set of recommendations D for a set of users G is defined as follows.

Definition 3 (Fairness). *Given a group G and a set of recommendations D, the fairness of D for G is defined as: $fairness(G,D) = \frac{\sum_{u \in G} fairness(u,D)}{|G|}$.*

Finally, we define the fairness-aware value of D for G as follows: $value(G,D) = fairness(G,D) \cdot \sum_{i \in D} relevanceG(G,i)$.

Aggregation Designs. We distinguish between the score-based and rank-based designs. In a *score-based design*, the prediction for an item is computed taking into account the relevance of the item for the group members. Firstly, we consider that strong user preferences act as a veto; this way, the predicted relevance of an item for the group is equal to the minimum relevance of the item scores of the members of the group: $relevanceG(G,i) = \min_{u \in G}(relevance(u,i))$. Alternatively, we focus on satisfying the majority of the group members and return the average relevance for each item: $relevanceG(G,i) = \sum_{u \in G} relevance(u,i)/|G|$.

In a *rank-based design*, we aggregate the group members recommendations lists by considering the ranks of their elements. Specifically, following the Borda

count method [2], each data item gets 1 point for each last place received in the ranking, 2 points for each next to last place, and so on, all the way up to k points for each first place received in the ranking. The item with the largest point total gets the first position in the aggregated list, the item with the next most points takes the second position, and so forth, up to locate the best k items. Overall, the points of each item i for the group G is computed as follows: $points(G, i) = \sum_{u \in G}(k-(p_u(i)-1))$, where $p_u(i)$ represents the position of item i in A_u.

Targeting at increasing the fairness of the resulting set of recommendations, we introduce also the *Fair* method, which consists of two phases. In the first phase we consider pairs of users in the group, in order to identify what to suggest. In particular, a data item i belongs to the top-k suggestions for a group G, if, for a pair of users $u_1, u_2 \in G$, $i \in A_{u_1} \bigcap A_{u_2}$, and i is the item with the maximum rank in A_{u_2}. For locating fair suggestions, initially, we consider an empty set D. Then, we incrementally construct D by selecting, for each pair of users u_x and u_y, the item in A_{u_x} with the maximum relevance score for u_y. If k is greater than the items we found using the above method, then we construct the rest of D, by serially iterating the A_u lists of the group members and adding the item with the maximum rank that does not exist in D.

Regardless of how similar the group members are, the first phase of the algorithm may yield few items (i.e., less than k). Moving to the second phase, we can assume a pseudo hierarchy inside the group members, meaning that the members that will be checked first, will have more relevant items for them, in the group list. So, by rearranging the order of the group members, we can influence the fairness achieved for each individual member. On the other hand, if we produce all top-k items from the first phase, then a number of items in the group list, may change accordingly to what order we examine the members. Again the vast majority of the items will be included, regardless of the members order. In both cases, the fairness of the list for the group does not change.

4 Experimental Evaluation

Dataset. In our experiments, we exploit 10.000 chimeric patient profiles [3] preserving the characteristics that exist in a real medical database. The patients health problems are described using the ICD10 ontology. Based on these profiles, we synthetically generated a document corpus and user ratings as follows. Initially, we generated *numDocs* documents, for each first level category of ICD10. For their corresponding keywords, we randomly selected $numKeyWords$ words from the description of the nodes in each subsequent subtree. We assume that all patients have given *numRatings* ratings. Specifically, we have divided the patients into three groups – *sparse*, *regular* and *dedicated*. The users in each group have given *few*, *average* and *a lot* of ratings, respectively. When ranking items based on human preferences, they tend to follow the power law distribution. To depict this, we have randomly selected *popularDocs* documents that will be the most popular. Given that patients are interested not only in documents

regarding their health, but also to some extent in others as well, for each patient, we have divided the ratings into *healthRelevant* and *nonRelevant*. Finally, for each rating generated in the previous step, we assigned randomly, *a value* in the range of 1 to 5. In our experiments, we set $numDocs = 270$, $numKeyWords = 10$ and $popularDocs = 70$.

Aggregation Methods. In our evaluation, we use the Minimum, Average, Borda and Fair designs. As a baseline, we will employ the Round-Robin aggregation design, considering each member of the group individually, and for each one, taking the item in his/her list with the highest score that does not already exist in the group list.

Evaluation Measures. For our experiments, to calculate the semantic similarity *SemS*, we use $w = 0.1$. To evaluate the similarity functions, we used the Mean Absolute Error (MAE) and the Root Mean Square error (RMSE). To quantify the success of each aggregation method, we compute the distance of each user's top-k recommendation list with that of the group. To calculate the final score, we take the average of those. For calculating the distance, we used the Kendall tau and the Spearman footrule distance.

Evaluation of Similarity Functions. To compare the two similarity functions, we focused on single users recommendations. In Fig. 1, we see the different values of MAE and RMSE for several k values. In all cases, the semantic similarity function gave better results than the rating function. This shows the added value of our solution on calculating effectively the similarity between users.

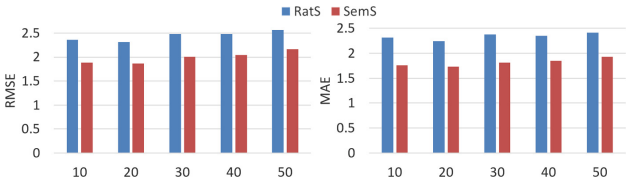


Fig. 1. The RMSE and MAE for different K

Evaluation of Aggregation Methods with Different Similarity Functions. To compute the distance between the top-k lists of the group members, and the group recommendation lists, we use the Kendall tau and Spearman footrule distances. We randomly selected 10 different groups that share the same *group similarity*. Group similarity is the similarity of all pairs of users in the group, averaged over the number of pairs. After generating group recommendations, we calculate for each member of the group the Kendall and Spearman distance. The distance score for the aggregation method is the averaged score of the summation of these distances over the number of group members. Following the same procedure for all 10 groups, the overall score for each aggregation method is the mean of the previously calculated scores, over the number of

different groups. To further supplement our findings, we compare the Kendall and Spearman distance for 10 different groups of size 5. The results are shown in Fig. 2 (a) and (b); *SemS* offers better results than *RatS*, regardless of the aggregation design used.

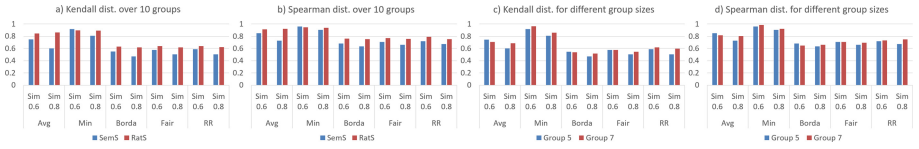


Fig. 2. The Kendall and the Spearman distance.

Evaluation of Aggregation Methods with Different Group Size. To get more accurate results, we study different sized groups, namely groups with 5 and 7 members. Using the Kendall distance (Fig. 2(c)), the rank-based methods give better results than the score-based methods. This is because the score-based methods consider the whole user's list, while the rank-based ones consider only the top-k items. The Minimum and Average designs, take into account the scores given to an item. For example, given one item, if a member of the group has a radically different relevance score for it than the rest, for Minimum, his opinion will act as veto, while in the case of Average, its group relevance will be brought down and might not make it into the group list. The rank-based methods are able to include more items from the members individual top-k recommendation lists, and hence give lower distances. Round-Robin is the worst method, while Borda and Fair have similar results. As expected, when the group similarity gets higher, all methods provide better results. Finally, the size of the group, given that we consider groups with the same similarity, slightly affects the quality of the results of the employed aggregation methods, and overall, the bigger the size of the group, the higher the Kendal tau distance. Figure 2(d) shows similar results for Spearman.

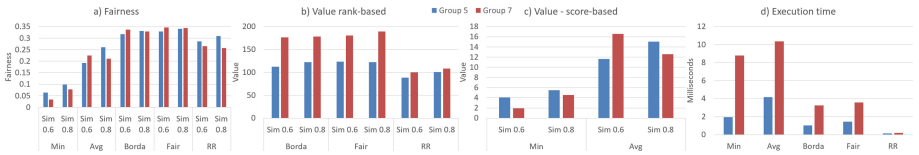


Fig. 3. Fairness, value and execution times.

Fairness. Since with fairness, we measure in essence how many of the items in an individual top-k list made it in the group recommendation list, which is inherently what we shown in Fig. 2(c) and (d), we expect complementary

results. In fact, the Minimum method that had the highest distance produces the lowest fairness. Although Borda gave marginally lower distance, Fair now gives better fairness. An explanation for this is that during the first phase of the Fair algorithm, we take into account items regardless of their relevance score – we actually want to include in the group list items that are relevant to most users, leading to higher fairness. Finally, the Round-Robin method gives lower fairness than the rest of the rank-based methods, because with Round-Robin, we do not consider the rest of the group members when constructing the recommendations list, but we consider each member individually from the others.

Value. Because of the inherent differences of the score-based and the rank-based designs, we cannot directly compare them regarding their value. To elaborate more, when we aggregate using the score-based techniques, for any given item, we directly compute its corresponding relevance score in the final group recommendation list. On the other hand, with the rank-based techniques, what we actually calculate is the rank that a specific item will have in the group recommendation list. But to find the value of a group recommendation list, we need the relevance score of all its items. Thus, we define the relevance score of an item in a group recommendation list, that is produced by a rank-based technique as the summation of its rank in each individual recommendation list of the group members. If an item is not present in a list, then it's score is 0. As it is apparent, we cannot directly compare score-based and rank-based aggregation methods. The score-based ones give a relevance score to an item in the range of [1,5], while the rank-based approaches, given that the group recommendation list consist of k items and the size of the group is s, give a relevance score in the range $[1, s * k]$. In Fig. 3(b), we see the *value* for the score-based aggregation methods. As expected, the Average method offers much better results. In Fig. 3(c), we compare the rank-based methods. These results complement those in Fig. 3(a). The Fair algorithm offers better overall value for the group recommendation list. The worst results are presented by Round-Robin; the fairness that Round-Robin offers is highly incidental and the relevance scores for those are low, since most users do not share the same items with the same high scores.

Execution Time. In Fig. 3(d), we show the time needed to aggregate the individual lists for each method. For each group size, we took randomly 10 different groups with the same group similarity. Average is the most time costly method. The time needed for Fair is marginally more, than the one needed for Borda. Nevertheless, the execution time is really small (at most 4 ms), offering better results (Fig. 3(a)).

5 Conclusions

In this work[2], we investigate how fairness can be modeled in group recommendations in the health domain. Specifically, we proposed a new similarity function

[2] The work was partially supported by the EU project iManageCancer (H2020, #643529), and the TEKES Finnish project Virpa D project.

that takes into account information provided by a patient's profile. As in modern Personal Health Systems [4], patient information is represented using standard terminologies, in this work, as a proof of concept, we employ the ICD10 ontology. Our experiments confirm that our proposed similarity function gives better results than traditional similarity functions based on ratings. We proceed even further, to explore and compare 4 different aggregation methods. Our experiments demonstrate the good behavior of our solution with respect to its target, i.e., to increase the fairness and utility of the suggested results.

References

1. Berg, G.M., Hervey, A.M., Atterbury, D., et al.: Evaluating the quality of online information about concussions. JAAPA **27**, 1547–1896 (2014)
2. Emerson, P.: The original borda count and partial voting. Soc. Choice Welf. **40**(2), 353–358 (2013)
3. Kartoun, U.: A methodology to generate virtual patient repositories. *CoRR*, abs/1608.00570 (2016)
4. Kondylakis, H., et al.: Digital patient: personalized and translational data management through the myhealthavatar EU project. In: IEEE EMBC, pp. 1397–1400 (2015)
5. Ntoutsi, E., Stefanidis, K., Nørvåg, K., Kriegel, H.-P.: Fast group recommendations by applying user clustering. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 126–140. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34002-4_10
6. Ntoutsi, E., Stefanidis, K., Rausch, K., Kriegel, H.P.: Strength lies in differences: diversifying friends for recommendations through subspace clustering. In: CIKM (2014)
7. Qi, S., Mamoulis, N., Pitoura, E., Tsaparas, P.: Recommending packages to groups. In: ICDM (2016)
8. Stratigi, M., Kondylakis, H., Stefanidis, K.: Fairness in group recommendations in the health domain. In: ICDE (2017)
9. Wiesner, M., Pfeifer, D.: Adapting recommender systems to the requirements of personal health record systems. In: IHI (2010)

Data Streams

Big Log Data Stream Processing: Adapting an Anomaly Detection Technique

Marietheres Dietz(✉) and Günther Pernul(✉)

Universität Regensburg, Lehrstuhl für Wirtschaftinformatik I, Universitätsstr. 31, 93053 Regensburg, Germany
{marietheres.dietz,guenther.pernul}@ur.de

Abstract. With the continuous increase in data velocity and volume nowadays, preserving system and data security is particularly affected. In order to handle the huge amount of data and to discover security incidents in real-time, analyses of log data streams are required. However, most of the log anomaly detection techniques fall short in considering continuous data processing. Thus, this paper aligns an anomaly detection technique for data stream processing. It thereby provides a conceptual basis for future adaption of other techniques and further delivers proof of concept by prototype implementation.

Keywords: Data stream · Anomaly detection · Log analysis
Real-time analysis

1 Introduction

Security incidents currently concern most of the companies as shown in a recent study [15]. In order to respond to security attacks before they gain a foothold, 81% of the enterprises nowadays apply log analyses [15], which can detect anomalous or even malicious behavior. Log analysis has been addressed by researchers for decades, while data stream research especially came to the fore as recently as the Big Data era. The combination of both issues, log analysis and Big Data stream processing, opens a new field of research, which has barely been addressed in literature to date. Although there have been early works on algorithms for anomaly detection in stream processing, data streams nowadays encompass Big Data dimensions [13], such as semi-structured formats, high volume and velocity - and thus generally require different processing systems [13] as well as different algorithms. This work addresses this research issue by developing a concept to adapt a statistical anomaly detection for log streams.

The remainder of this paper is organized as follows. The next section presents related concepts of previous works. Section 3 gives an overview of the developed concept for adapting the anomaly detection technique. Section 4 focuses the experiments conducted, including the implementation of the prototype and an evaluation of the practicability of the proposed solution. Section 5 outlines the main contributions and ways to enhance future research.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 159–166, 2018.
https://doi.org/10.1007/978-3-319-98812-2_12

2 Related Work

Since the beginning of the twenty-first century, log analysis and further anomaly detection techniques focused on preventing the newly emerged denial-of-service attacks, as presented by [19] amongst others. At that time, batch processing was the prevalent processing mode, which is demonstrated in the fundamental log analysis techniques described by [6].

However, some works already tackle the issue of developing anomaly detection techniques for data streams. One approach consists of applying sliding window to enable anomaly detection on data streams [1]. This method is the first known to support changes in baseline data instead of having a fixed set of historical data. The authors of [3] suggest an algorithm for anomaly detection in two-dimensional data streams. In another work, the current challenges in anomaly detection for data streams are stated [18]. More recently, the precedence of this topic regarding the advancing importance of continuously produced data is pointed out and the authors propose a theoretical concept for creating a framework for a Machine Learning technique in log stream analysis [11].

The work presented in this paper enlarges the state-of-the-art in various ways. First, the data streams differ from the vast majority of works as they represent log streams, which further possess a semi-structured format instead of structured or relational data. This adds complexity and addresses the Big Data dimension variety [14]. Second, previous works mainly focused on developing an anomaly detection algorithm or presenting a theoretical concept. This paper additionally shows an appliance in real-world scenarios and proves the practicability of the proposed concept by prototype implementation.

3 Conceptual Approach

The statistical technique for data stream anomaly detection as presented in this paper determines a baseline for normal instances and classifies all instances outside of the baseline as anomalies. The primal batch technique is presented in Sect. 3.1. Occurring issues of the technique due to the change in dynamics are stated in Sect. 3.2. Subsequently, Sect. 3.3 presents a solution addressing these issues and tailoring the primal technique to data streams.

3.1 The Primal Batch Anomaly Technique

The basic batch technique focuses on point-anomalies as these are the most common type of anomalies [5]. The adapted technique defines its baseline by a 95% confidence interval of numeric values referring to normal instances [6]. Such values can be an instance's attribute values or their aggregation, such as the times a system access to another system on an hourly basis. The confidence interval is computed by employing the following formulas [6]. At first, the arithmetic mean of the data instances x_i has to be defined.

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{1}$$

Next follows the calculation of the standard deviation.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}. \tag{2}$$

At last, the 95% confidence interval is computed [6].

$$[\mu - \frac{\sigma}{\sqrt{n}} \cdot 1,96; \mu + \frac{\sigma}{\sqrt{n}} \cdot 1,96]. \tag{3}$$

To address the issue of outdated data, new data should replace old data at regular intervals [6]. However, this is often manually managed [1]. A benefit that comes with the adaption, and thus the appliance of stream processing constructs, is that they can provide an implicit management of outphasing the oldest instances and absorption by the latest ones as described in Sect. 3.3.

3.2 Problem Statement

As the "store-then-process" strategy is no longer reasonable these days, data has to be processed on-the-fly, respectively on-line, which is considered as stream processing [9]. Streams consist of potentially infinite, continuously fast-moving data elements, also referred to as tuples [4], which further contain attribute values such as a timestamp [17]. Data stream management systems (DSMS) or stream processing applications (SPA) process streams with operators receiving tuples from incoming streams, employing a function on them and forwarding the processed tuples in outgoing streams [2]. As a result, processing on-the-fly implies statelessness of the stream and thus of the tuples. Thus, adapting the batch anomaly technique for real-time processing poses the following issues.

Problem 1: The determination of baselines is done by using previous instances, which arrive in the data stream before those instances investigated of being anomalies. As windows include real-time arriving instances, the investigated instances will emerge in the window, but have to be excepted for baseline determination.

Problem 2: The calculation of arithmetic mean (Formula 1) transforms the instances, which are no longer available for the calculation of the standard deviation (Formula 2).

3.3 The Proposed Stream Anomaly Technique

In order to manage stateful operations (e.g. aggregations of multiple tuple values), window constructs [8] are required. Figure 1 sketches the resulting concept. In detail, the two problems are addressed as follows.

Solution 1: To determine the baseline correctly, the latest instances in the window, which are checked for anomalies, have to be ignored or deleted (Fig. 1, step 1). Additionally, the investigated instances have to be maintained in a separate window (Fig. 1, step 5).

Solution 2: To solve the conflict between calculating the arithmetic mean and the part-aggregations for determination of the standard deviation simultaneously, clustering of attribute frequencies and separate storing of these clusters is required (Fig. 1, step 2):

$$a = x_i \forall i \in 1...n. \tag{4}$$

Subsequently, the mean can be calculated using the non-clustered version of the instances (Fig. 1, step 3). By combining both results, the standard deviation can be computed:

$$\sigma = \sqrt{\frac{\sum_{j=1}^{n}(a_j - \frac{1}{n}\sum_{i=1}^{n} x_i)^2}{n}}. \tag{5}$$

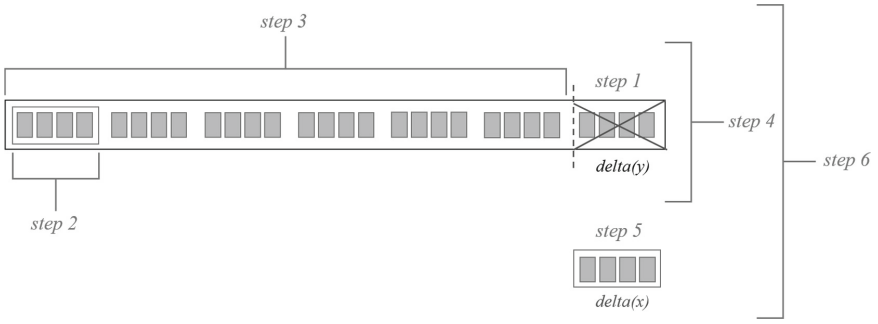


Fig. 1. Concept for adapting a batch anomaly detection for a log stream scenario

In general, two time-based windows build the developed concept's foundation, namely a sliding and a tumbling window. The tumbling window receives arriving instances within a certain timeframe *delta(x)*, e.g. 30 min, processes and deletes them afterwards before receiving instances within that timeframe again [7]. It is used to maintain the instances to be checked (Fig. 1, *delta(x)*). In contrast to tumbling windows, sliding windows do not delete all of their contained instances [7]. The applied sliding window receives instances within timeframe *delta(y)* (larger than *delta(x)*). It further slides by timeframe *delta(x)*, which means that after each processing the oldest instances are deleted while the newest arriving instances are instead incorporated [8]. This window is used to calculate the baseline (Fig. 1, *delta(y)*, step 4).

4 Experiments

The following experiments comprise a prototype implementation to realize the theoretical concept. Furthermore, a case study including a real-world firewall log stream serves experimentation and evaluation of the prototype.

4.1 Implementation

The prototype development process is oriented on the work of [16]. It consists of a project developed in Java with the stream processing system Apache Flink[1]. The logs addressed in this paper are produced by a firewall recording occurring events, which are also known as event logs [12]. This data stream source, the firewall, lies within the enterprise infrastructure provided by the DINGfest[2] project. Thereafter, the produced log stream enters the buffer technology Apache Kafka[3] where it is transmitted to Apache Flink. In Apache Flink, the prototyped concept is applied to the stream. The results of this analysis are printed in the Flink console or visualized. To enable visualization of the current frequencies and the baseline values, the streams are conveyed to the visualization system Grafana[4]. The transfer of the stream instances is done via influxdb[5], which is a database that is able to transfer time series data points to Grafana.

4.2 Evaluation

The experimental assessment of this work is carried out by evaluating the prototype through controlled environments within a case study as proposed by [10]. The case study of this work encompasses a log stream describing real-world security events of the respective system, which produces data instances in milliseconds to seconds. Hence the analyzed stream corresponds to the Big Data dimensions in velocity and volume. The analyzed log stream possesses a semi-structured stream schema containing security-relevant information of a firewall, which contains network data instances.

Sample output line of the Flink console presenting the anomaly detection result for the case study

```
2> anomaly: false || checked tuple(udp) : start: 02/02/2018
08:00:00; count: 8 || confidence interval(udp) :  end: 02/02/2018
08:00:00; interval: [6;17]
```

(Extract of the anomaly detection in frequencies in the firewall protocol attributes)

The above shows a sample output line of the frequencies of the firewall protocol attributes in the Flink console. At the beginning a boolean value displays whether the analyzed attribute frequency is an anomaly (*true*) or not (*false*). Afterwards, the characteristics of the checked window are listed, including the attribute value, e.g. *udp*, the actual frequency and the start time of the window. In the end, the baseline is described by the attribute value, the end of the time frame of the baseline window and the confidence interval. Thereby the

[1] https://flink.apache.org.
[2] A project funded by the German Ministry of Education and Research that provides real-world log streams of various systems within an enterprise infrastructure.
[3] https://kafka.apache.org.
[4] https://grafana.com.
[5] https://www.influxdata.com.

attribute values coincide as well as the start time of the checked window with the end time of the baseline window. Thus, the fact that the baseline calculation happens independent of the checked frequencies is confirmed.

Table 1 depicts sample results of the case study. It enlarges the first analysis of the protocol attribute by showing the concrete access frequencies from internet protocol (IP)-addresses to other IP-addresses, which each refer to various systems within the enterprise infrastructure. In more detail, the source IP belongs to the system that tries to or accesses another system. While the frequencies of the protocol attributes can be analyzed at first glance, the anomaly detection of the IP-address frequencies allows an additional in-depth view of occurring anomalies.

Table 1. Sample output presenting the anomaly detection result for the firewall IP-address access frequencies in the case study

Anomaly	Attribute	Attribute value	Current access frequency	Baseline access frequency
False	protocol	udp	8	[6;17]
False	source IP	194.150.241.49	8	[8;9]
False	destination IP	194.150.241.55	8	[8;9]

The graphical interface for visualization of the results is accomplished with a Grafana dashboard. Figure 2 shows a panel, in which the frequencies of the *udp* protocol type against the time are plotted. Anomalies become apparent when the line or point representing the frequency of the attribute value exceeds the line or point of the upper bound of the confidence interval or falls below the lower bound.

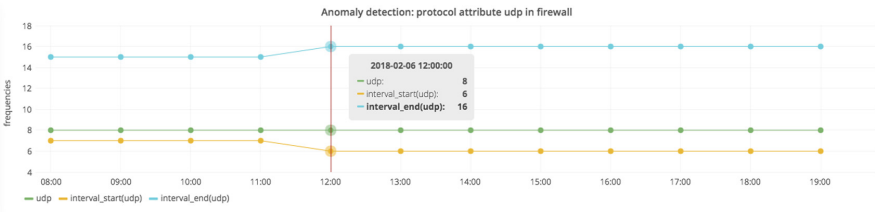


Fig. 2. Sample output of the dashboard in anomaly detection of the frequencies in the protocol attribtute *udp*

In the following the issues and solutions of the approach as well as resulting outcomes are discussed. Generally, one circumstance hampers the anomaly detection in this work: Events or attribute values occurring rarely pose an issue,

as, if a too small capture period of the baseline is chosen, the confidence interval will feature a very small width. The baseline start is further close to zero, as in most of the times of the baseline time frame the attribute value will not emerge. This situation is observed in the case study in the case of the protocol attribute value *tcp*, which does not emerge on some days and occurs on few points in time on other days. However, this issue is diminished by slightly adjusting the calculation of the arithmetic mean within the baseline calculation. By only considering frequencies greater than zero as well as the times such frequencies occur, the arithmetic mean no longer converges as strongly towards zero as before.

In other respects, there are no issues with sufficiently large continuous data streams and a correspondingly broad period of capture. Thus, the statistical anomaly detection of this work is thus a suitable way to detect outliers in Big log data streams.

5 Conclusion and Future Work

The work's general contribution is a concept for adapting batch processing technique for deployment in a streaming context. An additional contribution is given by solving the adaption with windows, where the calculated baseline is based on a defined data period *delta(y)*, e.g. six weeks, which means that only the tuples that appeared from *t0* until *t0 + delta(y)* are considered. From this follows that change in frequencies is minded, with the result that it remains valid for future scenarios. Although other statistical anomaly methods are not explicitly considered, underlying parts of the proposed concept can be applied in similar scenarios, e.g. in classification algorithms for anomaly detection purposes, which are also based on declaring a larger set of instances as training set and investigating the latest instances (test set) of being anomalies.

As the evaluation suggests, the timeframe for baseline calculation influences the number of detected anomalies. Future work may therefore consider customization of the timeframe for baseline calculation. For instance, an implemented user interaction with the dashboard can enable a broader group of users to customize the timeframe of the baseline window by evaluating the different effects visually.

Acknowledgements. Part of this research was supported by the Federal Ministry of Education and Research, Germany, as part of the BMBF DINGfest project (https://dingfest.ur.de).

References

1. Agarwal, D.: Detecting anomalies in cross-classified streams: a Bayesian approach. Knowl. Inf. Syst. **11**(1), 29–44 (2006)
2. Andrade, H.C.M., Gedik, B., Turaga, D.S.: Fundamentals of Stream Processing: Application Design, Systems, and Analytics. Cambridge University Press, Cambridge (2014)

3. Angiulli, F., Fassetti, F.: Detecting distance-based outliers in streams of data. In: Silva, M.J., et al. (eds.) Proceedings of the 2007 ACM Conference on Information and Knowledge Management, p. 811. ACM, New York (2007)
4. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Abiteboul, S. (ed.) Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 1–30. ACM, New York (2002)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection. ACM Comput. Surv. **41**(3), 1–58 (2009)
6. Chuvakin, A., Schmidt, K., Phillips, C.: Logging and Log Management: The Authoritative Guide to Dealing with Syslog, Audit Logs, Events, Alerts and other IT 'Noise'. Elsevier Science, Burlington (2012)
7. Cugola, G., Margara, A.: Processing flows of information. ACM Comput. Surv. **44**(3), 1–62 (2012)
8. Golab, L., Özsu, M.T.: Issues in data stream management. ACM SIGMOD Rec. **32**(2), 5–14 (2003)
9. Hébrail, G.: Data stream management and mining. In: Fogelmann-Soulié, F., Perrotta, D., Piskorski, J., Steinberger, R. (eds.) Mining Massive Data Sets for Security, pp. 89–102. IOS Press (2008)
10. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
11. Hussain, A.R., Hameed, M.A., Fatima, S.: A proposal: high-throughput robust architecture for log analysis and data stream mining. In: Saini, H.S., Sayal, R., Rawat, S.S. (eds.) Innovations in Computer Science and Engineering. AISC, vol. 413, pp. 305–314. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-0419-3_36
12. Kent, K., Souppaya, M.P.: Guide to computer security log management. Technical report, National Institute of Standards and Technology, Gaithersburg, MD. https://www.nist.gov/publications/guide-computer-security-log-management
13. Korolov, M.: Log management is leading use case for big data (2015). https://www.csoonline.com/article/2935362/data-protection/log-management-is-leading-use-case-for-big-data.html
14. National Institute of Standards and Technology: Big data interoperability framework: Volume 1, defintions. https://bigdatawg.nist.gov/_uploadfiles/NIST.SP.1500-1.pdf
15. Neely, L.: 2017 threat landscape survey: users on the front line. Technical report, SANS Institue. https://www.sans.org/reading-room/whitepapers/threats/2017-threat-landscape-survey-users-front-line-37910
16. Nunamaker, J.F., Chen, M.: Systems development in information systems research. In: Twenty-Third Annual Hawaii International Conference on System Sciences, pp. 631–640. IEEE Computer Society Press (1990)
17. Olbrich, S.: Warehousing and analyzing streaming data quality information. In: AMCIS 2010 Proceedings (2010)
18. Sadik, S., Gruenwald, L.: Research issues in outlier detection for data streams. ACM SIGKDD Explor. Newslett. **15**(1), 33–40 (2014)
19. Winding, R., Wright, T., Chapple, M.: System anomaly detection: mining firewall logs. In: 2006 SecureComm and workshops, pp. 1–5. IEEE, Piscataway, NJ (2006)

Information Filtering Method for Twitter Streaming Data Using Human-in-the-Loop Machine Learning

Yu Suzuki(✉) and Satoshi Nakamura

Nara Institute of Science and Technology,
8916-5 Takayama, Ikoma, Nara 6300192, Japan
ysuzuki@is.naist.jp

Abstract. There are a massive amount of texts on social media. However, only a small portion of these texts is informative for a specific purpose. If we accurately filter the texts in the streams, we can obtain useful information in real time. In a keyword-based approach, filters are constructed using keywords, but selecting the appropriate keywords to include is often difficult. In this work, we propose a method for filtering texts that are related to specific topics using both crowdsourcing and machine learning based text classification method. In our approach, we construct a text classifier using FastText and then annotate whether the tweets are related to the topics using crowdsourcing. In this step, we consider two strategies, optimistic and pessimistic approach, for selecting tweets which should be assessed. Then, we reconstruct the text classifier using the annotated texts and classify them again. We assume that if we continue instigating this loop, the accuracy of the classifier will improve, and we will obtain useful information without having to specify keywords. Experimental results demonstrated that our proposed system is effective for filtering social media streams. Moreover, we confirmed that the pessimistic approach is better than the optimistic approach.

1 Introduction

A massive amount of texts are on social network services, and much information about real situations can be gleaned from these texts. There have been many studies on how to extract the necessary information from these data [1]. Information filtering [2] is a process for retrieving texts from text streaming, and the keyword-based method is often used for this. However, it is difficult to set appropriate keywords that correspond to the information needed, because of two reasons; information need are generally vague and sentences on social networks are always broken. Crowdsourcing is one possible solution to solve these issues. However, assessing all tweet streaming data by crowdsourcing is unrealistic, because there are a large amount of tweets, then we should pay many wages to the workers for assessing all tweets. To reduce the amount of crowdsourcing tasks, we used machine learning.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 167–175, 2018.
https://doi.org/10.1007/978-3-319-98812-2_13

In our research, we have developed a system that collects a small number of texts relevant to subjective queries from among a large text stream by using machine learning and crowdsourcing. Our objective is to extract tweets related to topics from text streaming. The topics we assume are general, and the extracted tweets are worthwhile for many users (i.e., not personalized topics).

To achieve our objective, we combine two techniques; crowdsourcing and machine learning, which is called Human-in-the-Loop. In this research, we investigated a problem that occurs when active learning is performed on information filtering with regard to the tweets presented to the workers. Information filtering was performed using machine learning and crowdsourcing so as to determine the accuracy and the cost involved in obtaining relevant tweets.

2 Related Work

In social network services, numerous useful texts are posted, such as those related to the spread of influenza or the occurrence of an accident. Several systems have been proposed to capture these incidents quickly [3] and thereby to utilize social media services as a kind of social sensor. Many information filtering methods to glean useful information from streaming data have been proposed, and many of them are used in systems generated for personalization [4] in what is called "personalized information filtering." These techniques are based on information retrieval and are not appropriate for short texts such as tweets. Therefore, bag-of-words features along with domain-specific knowledge [5], the relationship between users [6], and user behaviors such as re-tweeting [7] are used as features to filter tweets.

Relevance feedback is important for improving the accuracy of information filtering. Rocchio [8] proposed a relevance feedback mechanism on the vector space model. In this mechanism, since more accuracy is improved as more feedback is given, a method using crowdsourcing for feedback has been proposed. [9,10]. However, in these methods, it is assumed that the set of documents that are compatible is sufficiently large as compared with the whole set, so it was not clear whether it can be used for information filtering. In this research, we clarify whether relevance feedback by crowdsourcing is effective when the relevant document is extremely few.

3 Information Filtering Method

Our proposed information retrieval system uses a combination of crowdsourcing and machine learning techniques. An overview is shown in Fig. 1.

3.1 Building Classifier

We use FastText [11,12], an application for word embedding and classification, for classifying tweets. With FastText, we input a training set, a pre-trained embedding model, and parameters.

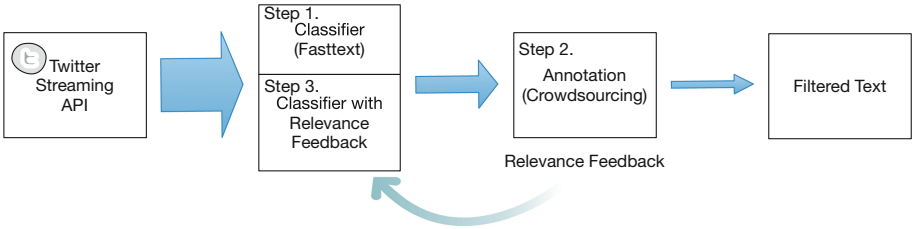


Fig. 1. Overview of the proposed system.

First, we prepare a dataset for constructing an initial classifier. We remove tweets from the dataset if they include URLs or mention other users. Then, we prepared a set of tweets that are related to specific topics using crowdsourcing. Each tweet is given the label *positive* or *negative*. The label *positive* means that the tweets are relevant to the topics, and the label *negative* means that the tweets are irrelevant to the topics. At this time, the classifier calculates the certainty.

Next, we extract bag-of-word features from the texts. MeCab[1] with the IPADIC-Neologd dictionary[2] was used as a morphological analyzer. There are typically many newly coined words in the tweets, so we use a dictionary that includes new words. Then, we extract nouns, verbs, adjectives, and adverbs. We use a pretrained embedding model[3] constructed using the Japanese Wikipedia copus[4]. This enables synonyms to be handled in the same way.

Finally, using the classifier included in FastText implementation [13], we construct a classifier of the tweets. We assigned a *positive* label or a *negative* label to the unlabeled tweets.

3.2 Classify Tweets

Next, we classify the tweets obtained from text streaming by using the classifier described in Sect. 3.1. We classify the new tweets and continue classification until we can look for tweets labeled as *positive*. The tweets judged as positive by the classifier perform annotation, which is the next step. Tweets that are judged as negative are discarded after this step.

3.3 Annotation

As shown in Fig. 2, we manually judged whether or not the tweets classified as positive by the classifier were positive. Although accuracy can be improved by making judgments (and the more people, the better the accuracy), the accuracy of the final classifier improves as the number of judged tweets increases, even if

[1] http://taku910.github.io/mecab/.
[2] https://github.com/neologd/mecab-ipadic-neologd.
[3] https://qiita.com/Hironsan/items/513b9f93752ecee9e670.
[4] https://dumps.wikimedia.org/jawiki/20170101/.

the accuracy is low. Each judgment made was considered final (i.e., there was no redundancy by multiple workers or averaging that took place).

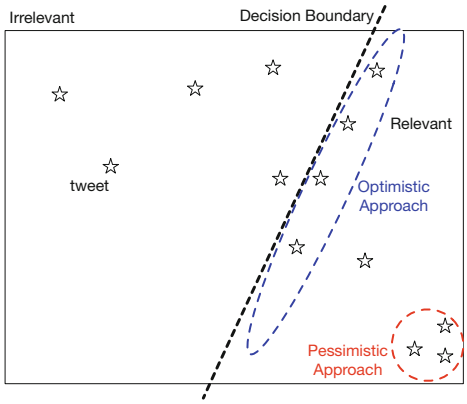


Fig. 2. Intuitive example of our proposed method. (Color figure online)

Interestingly, the performance of the classifier changed depending on which text was judged by a worker. Campbell et al. [14] pointed out that there are differences in the performance of classifiers depending on the sample selection method. The following two ideas were considered as an optimal approach and a pessimistic approach.

Strategy 1: Optimistic Approach. When a classifier classifies a tweet, it is a way to prioritize tweets that are difficult to judge as positive or negative. Lewis et al. [15] proposed a technique called uncertainty sampling, which Simon et al. [16] later used in an application using support vector machines. In this method, tweets near a decision boundary are annotated. The blue part in Fig. 2 shows the tweets which should be annotated using this approach.

Strategy 2: Pessimistic Approach. In this strategy, a presentation is made to the worker in descending order of probability of what the classifier judges as positive. In the information filtering handled in this research, many tweets are considered to be unrelated, which means that even if the positive probability of the classification result is high, it is unlikely to be positive. Therefore, this method was devised to judge, as accurately as possible, the appropriate tweet. The red portion in Fig. 2 shows the tweets which should be annotated using this approach.

This method is considered suitable for when the classification performance by classifiers is not sufficient for the unlabeled tweet.

Table 1. Parameters for FastText.

Parameter	Value
Number of epochs	10,000
Size of vectors	300
Number of buckets	100,000,000
Loss function	Negative sampling
Number of negatives sampled	10
Minimum number of word occurrences	1
Max length of word n-gram	1
Learning rate	0.075

Table 2. Experimental setting.

	Optimistic	Pessimistic
Manually processed tweets	94,597	176,238
No. of workers	72	72

4 Evaluation

We used the optimistic and pessimistic approaches (Sect. 3.3) to determine the effectiveness of these strategies using the number of adequate tweets to be obtained (Table 2).

Data. We prepared two groups of tweet data: labeled and unlabeled. As labeled data, we used all 3,580 manually collected tweets. As unlabeled data, the Twitter Streaming API was used to collect more than 1 million tweets in advance. To ensure the same conditions when making a comparison of the two strategies, the tweets to be categorized were made identical for both strategies. If the performance was the same, the same tweet was extracted.

Procedure. Evaluation experiments were carried out as follows.

- Build a classifier using labeled data.
- Arrange unlabeled data in chronological order and classify using a classifier. Obtain 1,000 positive tweets.
- Classify tweets against positive tweets by crowdsourcing.
- Add judgment result of crowdsourcing to labeled data and return to 1.

First, we gathered tweets by using the optimistic strategy and then collected tweets again by the pessimistic strategy. The number of workers was the same in both strategies, but the workers themselves were different.

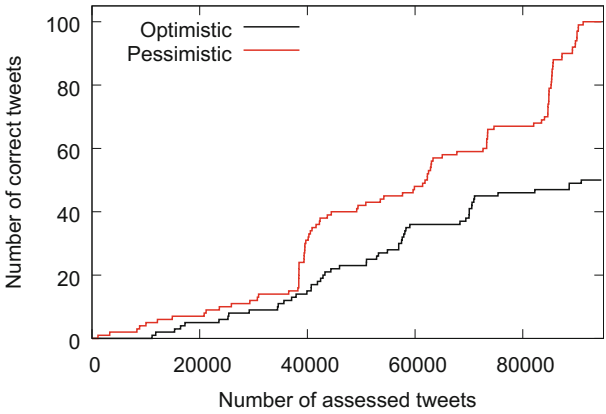


Fig. 3. Ratio of positive tweets and correct answer rate.

Results and Discussion. The hyperparameter used for the classifier is shown in Table 1. In a preliminary experiment using initially labeled data, parameters that can classify positive and negative with high accuracy were obtained by grid search.

We obtained 94,598 and 176,238 assessments by using the optimistic and pessimistic approaches, respectively. For comparison, we used all 94,548 assessments by the optimistic approach and the first 94,548 assessments by the pessimistic approach. The results are shown in Fig. 3. We discovered that the correct tweet could be collected twice as fast by the pessimistic approach than the optimistic approach. Specifically, when the number of the assessed tweets was 40,000, the system could collect many correct tweets when it exceeded 85,000.

Figure 4 shows the number of model reconstruction (steps) vs. the ratio of positive tweets. In this figure, a point shows how much percentage of new tweets classifier judged as positive at a step. For example, if the value at step 3 is 0.1, the classifier judges the tweets as positive at the ratio of 1 to 10 when the classifier was rebuilt three times. From this figure, in the pessimistic approach, many tweets were judged as positive in the first three steps, but after step 4 the ratio is lower than the optimistic approach and the ratio is converged to about 0.02. On the other hand, in the optimistic approach, in the first step, the classifier judge a small number of the positive tweet than that by the pessimistic approach. However, the ratio of positive tweets does not decrease in subsequent steps. The same number of positive tweets are selected at each level. Therefore, decreasing the ratio of positive tweets means that it is possible to judge many tweets in a short time. As a result, it was found that the pessimistic approach can handle many tweets as compared with the optimistic approach.

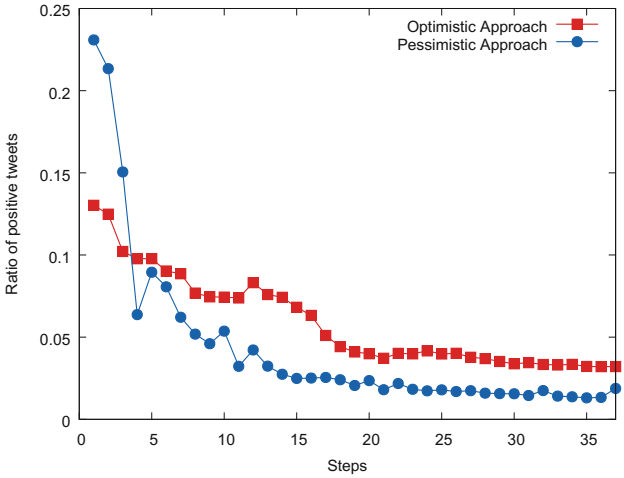


Fig. 4. The number of steps vs. ratio of positive tweets

5 Conclusion

In this paper, we proposed a method for filtering tweet streams using both crowdsourcing and machine learning. In this research, we investigated a problem that occurs when active learning is performed on information filtering with regard to the tweets presented to the workers. Information filtering was performed using machine learning and crowdsourcing so as to determine the accuracy and the cost involved in obtaining relevant tweets.

In the evaluation experiment, we confirmed how many relevant tweets the system can collect when combining crowdsourcing and FastText, a machine learning based classifier. At this time, we compared two strategies, optimistic and pessimistic approach, to select tweets presented to workers which are useful for improving the accuracy of the classifier. As a result, we were able to collect relevant tweets as much as 50 tweets and 100 tweets for optimistic approach, and pessimistic approach for 4,500 JPY (45 USD), respectively. At this time, we got an assessment result of around 95,000. We were able to obtain unfavorable tweets with keywords so that we could show the usefulness of the proposed method.

In future work, we should combine the existing keyword-based approach with our proposed crowdsourcing and machine learning based approach for constructing more accurate information filtering.

Acknowledgments. The research results have been achieved by "Research and Development on Fundamental and Utilization Technologies for Social Big Data," the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN.

References

1. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI 2007, pp. 2670–2676, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2007)
2. Belkin, N.J., Croft, W.B.: Information filtering and information retrieval: two sides of the same coin? Commun. ACM **35**(12), 29–38 (1992)
3. Abel, F., Hauff, C., Houben, G.J., Stronkman, R., Tao, K.: Twitcident: fighting fire with information from social web streams. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012 Companion, pp. 305–308, New York. ACM (2012)
4. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating “word of mouth”. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1995, pp. 210–217, New York. ACM Press/Addison-Wesley Publishing Co. (1995)
5. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M.: Short text classification in twitter to improve information filtering. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR 2010, pp. 841–842, New York. ACM (2010)
6. Hannon, J., Bennett, M., Smyth, B.: Recommending twitter users to follow using content and collaborative filtering approaches. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys 2010, pp. 199–206, New York. ACM (2010)
7. Uysal, I., Croft, W.B.: User oriented tweet ranking: a filtering approach to microblogs. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM 2011, pp. 2261–2264, New York. ACM (2011)
8. Rocchio, J.J.: Relevance feedback in information retrieval. In: Salton, G. (ed.) The Smart Retrieval System: Experiments in Automatic Document Processing, pp. 313–323. Prentice Hall (1971)
9. Grady, C., Lease, M.: Crowdsourcing document relevance assessment with mechanical turk. In: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, CSLDAMT 2010, pp. 172–179, Stroudsburg, PA, USA. Association for Computational Linguistics (2010)
10. Alonso, O., Baeza-Yates, R.: Design and implementation of relevance assessments using crowdsourcing. In: Clough, P., et al. (eds.) Advances in Information Retrieval, pp. 153–164. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20161-5_16
11. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
12. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 427–431. Association for Computational Linguistics (2017)
13. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext.zip: compressing text classification models, December 2016
14. Campbell, C., Cristianini, N., Smola, A.J.: Query learning with large margin classifiers. In: Proceedings of the Seventeenth International Conference on Machine Learning, ICML 2000, pp. 111–118, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2000)

15. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: Croft, B.W., van Rijsbergen, C.J. (eds.) SIGIR 1994, pp. 3–12. Springer, New York (1994). https://doi.org/10.1007/978-1-4471-2099-5_1
16. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. J. Mach. Learn. Res. **2**, 45–66 (2002)

Parallel n-of-N Skyline Queries over Uncertain Data Streams

Jun Liu[1,2], Xiaoyong Li[1,2(✉)], Kaijun Ren[1,2], Junqiang Song[1,2], and Zongshuo Zhang[3]

[1] College of Meteorology and Oceanology, National University of Defense Technology, Changsha, China
{liujun12a,sayingxmu,renkaijun,junqiang}@nudt.edu.cn
[2] College of Computer, National University of Defense Technology, Changsha, China
[3] School of Engineering, University of Central Lancashire, Preston, England
ZZhang21@uclan.ac.uk

Abstract. The skyline query over uncertain data streams has attracted considerable attention recently, due to its significance in helping users analyze big data. However, existing uncertain skyline queries with sliding window model only focus on retrieving the most recent N streaming items, which limits the query flexibility and efficiency. In this paper, we propose an efficient parallel method for processing uncertain n-of-N skyline queries. Specifically, we define the parallel uncertain skyline queries with n-of-N model, and propose a novel parallel query framework. Moreover, we propose a sliding window partitioning strategy, as well as a streaming items mapping strategy to realize the load balance. Additionally, we provide an encoding interval technique to further improve the query efficiency. Extensive experiments are conducted to demonstrate the effectiveness and efficiency of our proposals.

Keywords: Uncertain data · Parallel skyline queries · n-of-N model

1 Introduction

The skyline query is also referred as Pareto optimal query [1], which has widely used in various domains like market analysis, multi-target optimization, and preference query. Generally, users may need the query results of different query scopes at the same time. Unfortunately, it is difficult to address the problem above because the traditional skyline queries only focus on the most recent N streaming items. The skyline query with n-of-N model provides an effective way to enhance the query flexibility, by supporting the query for the most recent $n\,(n \leq N)$ items at the same time. Nevertheless, compared with the traditional uncertain skyline queries with full window model, the n-of-N skyline query over uncertain data streams faces the following challenges:

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 176–184, 2018.
https://doi.org/10.1007/978-3-319-98812-2_14

- The computation of the uncertain n-of-N skylines not only needs to examine the dominance relationships, but also requires to compute the skyline probabilities. Therefore, the skyline computation has a demand for powerful processing capability.
- The continuous items arriving makes the number of items concerned by users more large, but the response time more limited. Therefore, the existing centralised methods will be difficult to meet the query requirements.

In this paper, we extensively study the parallel approach for computing the uncertain skylines against n-of-N model. To summarize, we provide the following contributions:

- We define the parallel n-of-N skyline queries over uncertain data streams;
- We propose a novel parallel query framework and two strategies to realize the load balance between each node;
- To further improve the query efficiency, we provide an encoding interval scheme to transform the n-of-N query into stabbing query;
- We conduct extensive experiments with real deployment under different parameter settings to verify the effectiveness and efficiency of our proposals.

The rest of this paper is organized as follows: Sect. 2 reviews the related work. Section 3 describes the definitions and notations. Section 4 describes the parallel query model. Section 5 elaborates our parallel query algorithms. We extensively evaluate the performance of our proposals in Sect. 6. Finally, we conclude the paper and outline our future work in Sect. 7.

2 Related Work

Since its introduction [2], uncertain skyline queries have received considerable attention in database community in recent years. Particularly, Yang et al. [3] present an uncertain dynamic skyline (UDS) query and propose several effective pruning strategies. In addition, He et al. [4] propose three effective pruning techniques to obtain the skyline in the interval. To process skyline queries efficiently, De Matteis et al. [5] propose a parallelization algorithm to achieve near-optimal speedup with several load-balancing strategies. Moreover, Park et al. [6] propose an efficient parallel algorithm $SKY\text{-}MR^+$ for processing skyline queries.

Unlike the skyline queries with full-window model, n-of-N skyline queries over uncertain data streams focus the attention on different size windows, which concern the most recent $n\,(n \leq N)$ data items. Typically, Yang et al. [7] propose an uncertain n-of-N skyline query method which uses RDO tree, interval tree and time-label tree to optimize the update of data streams. Similarly, an effective pruning technique is proposed in [8] to improve the query efficiency by minimizing the number of uncertain elements to be kept.

3 Definitions and Notations

Definition 1 (*dominance* [9]). *For any two items a and b from the d-dimensional space, a dominates b, denoted by $a \prec b$, iff $a.i \leq b.i$ for all dimensions $1 \leq i \leq d$ and there exists a dimension j satisfying $a.j < b.j$.*

Definition 2 (*skyline probability* [10]). *The skyline probability of an uncertain item e in DS_N is defined by*

$$P_{sky}(e) = P(e) \times \prod_{e' \in DS_N, e' \prec e} (1 - P(e')) \tag{1}$$

We also define the $P_{new}(e)$ with items arriving later than e, that is

$$P_{new}(e) = P(e) \times \prod_{e' \in DS_N, e' \prec e, k(e') > k(e)} (1 - P(e')) \tag{2}$$

Definition 3 (*candidate set* $S_{N,q}$ [9]). *We use $S_{N,q}$ to denote the subset $\{e | e \in DS_N\}$ in which the item e satisfying $P_{new}(e)$ not small than q.*

Definition 4 (*n-of-N model* [7]). *N represents the sliding window size and $n\,(n \leq N)$ is the most recent items. Particularly, the sliding window with size N is just a special case of $n = N$.*

Definition 5 (*uncertain n-of-N skylines*). *The uncertain n-of-N skylines is a subset of DS_N, which includes items whose $P_{sky}(e)$ within the most recent n items are not small than q.*

For brevity, the frequently used notations in the paper are summarised in Table 1.

Table 1. Frequently used notations

Symbol	Meaning
U	The number of streaming items arrived
W	The global sliding window
$\|W\|$	The size of the sliding window W
W_i	The local sliding window
$k(e)$	The item e arrives $k(e)$-th in DS
e_{new}	The new arriving streaming item in W
e_{old}	The expired streaming item in W
a_e	The youngest item (but older than e) dominating e
$P(e)$	The uncertainty of item e

4 Parallel Query Model

4.1 Query Interval Encoding

To calculate the uncertain n-of-N skylines, we need to maintain a candidate set and map the items in it to the stabbing query intervals according to the method proposed by Lin et al. Thus if item e satisfies $U - n + 1 \in (k(a), k(e)]$, then it is a skyline object of DS_N.

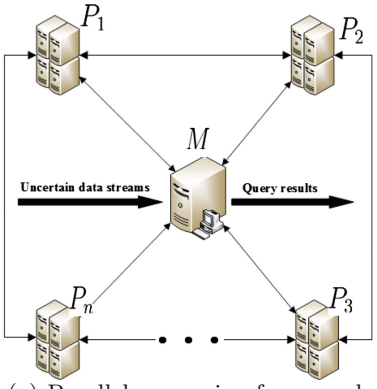
Definition 6 (*stabbing query interval*). *To process the query efficiently, we use the red-black tree RBI to organize stabbing query intervals. A semi-closed query interval is denoted by a structure Inv, which contains three attributes including item e, left endpoint l and right endpoint r (i.e., the time label $k(e)$) of the interval. The left endpoint of the interval for e is calculated as follows:*

- *For each item $e \in S_{N,q}$, we assume there are m items dominating and arriving earlier than e, and use $a_1, a_2, ..., a_m$ in sequential order to denote them.*
- *For $\forall e \in S_{N,q}$, let $j = min\{k | 1 \leq k \leq m \wedge P_{new}(e) \times \prod_{i=k}^{m}(1 - P(a_i)) \geq q\}$.*
- *If j exists, there is $l = 0$ when $j = 1$ or $l = k(a_{i-1})$ when $j > 1$;*
- *If j does not exist, there is $l = 0$ when $m = 0$ or $l = k(a_m)$ when $m \neq 0$.*

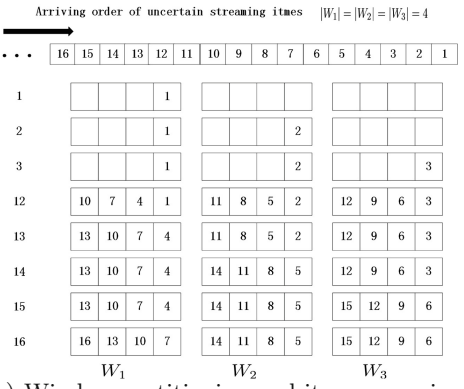
4.2 Parallel Query Framework

In this paper, our proposals mainly obtain the query results with the parallel iterative processing on each local sliding window. Figure 1(a) shows our parallel processing framework, which contains two kinds of nodes:

- Monitor node (M): the node is responsible for maintaining W, mapping new items to compute nodes, and collecting query results.
- Compute node (P_i): the nodes are used to compute the stabbing query interval of e_{new} and process n-of-N skyline query.



(a) Parallel processing framework



(b) Window partitioning and items mapping

Fig. 1. Parallel query model.

4.3 Sliding Window Partitioning and Streaming Items Mapping

In this paper, we assume all compute nodes have the same calculated capacity and P_i maintains $W_i\,(1 \leq i \leq n)$. Therefore, we partition the global sliding window averagely by $|W_i| = |W|/n$. In order to improve the efficiency of the dominance tests between items, we use a R tree RST to organize all streaming items. Moreover, to achieve the load balance between compute nodes, we map the new arriving items to the compute nodes by an alternate mapping strategy, such as first to P_i, then to P_{i+1}, and this is followed by P_{i+2}, ..., P_i. As shown in Fig. 1(b), we first transmit e_1 to P_1, and then e_2 to P_2, followed by e_3 to P_3, and so forth.

5 Parallel Query Algorithms

5.1 Parallel Query Processing

Assume that the initialization is complete and W is full, then our proposed parallel query framework can be summarized in Algorithm 1.

Algorithm 1. Parallel Uncertain n-of-N Skyline Queries

```
   Input: the uncertain data streams
   Output: the uncertain n-of-N skylines
1  begin
2      while there comes a new item e_new to M do
3          M maps e_new to P_i according to the alternate mapping strategy;
4          P_i compute the stabbing query interval of e_new with Algorithm 2;
5          foreach compute node P_j (1 ≤ j ≤ n) do
6              update the S_{N,q} it maintains with Algorithm 3;
7              stab RBI by U − n + 1 and return query results to M;
```

In Algorithm 1, M first transmits e_{new} to compute node P_i (Line 3). Then, P_i computes the stabbing query interval of e_{new} with other compute nodes (Line 4). Furthermore, each compute node updates the stabbing query intervals of all items dominated by e_{new} (Lines 5–6). Finally, each compute node stabs the RBI it maintains by $U - n + 1$, and sends the query results to M (Line 7).

5.2 Computing e_{new} Interval

When e_{new} arrives, we compute its query interval with items dominating it. In Algorithm 2, P_i first gets all items dominating e_{new} and computes the query interval of e_{new} with items above (Lines 3–7). Then, P_i puts all items dominating e_{new} and satisfying $P_{sky}(e_{new}) \geq q$ into the dominating set of e_{new} (Line 8).

Algorithm 2. Computing e_{new} Stabbing Query Interval

Input: the item e_{new}; the P_i with e_{new}
Output: the stabbing query interval of e_{new}

```
1 begin
2     P_new(e_new) := 1;
3     P_i finds items dominating e_new against W_i;
4     P_i transmits e_new to other compute nodes P_j (1 ≤ j ≤ n, j ≠ i);
5     foreach compute node P_j (1 ≤ j ≤ n, j ≠ i) do
6         find all items dominating e_new against W_j and send them to P_i;
7     P_i compute the stabbing query interval of e_new with Definition 6;
8     compute the dominating set of e_new;
```

5.3 Maintaining $S_{N,q}$

In Algorithm 3, $BD(e_{new})$ is the set including items dominated by e_{new}. If there is $P(e_{new}) > (1-q)$, there will be $P_{new}(e) < q$ absolutely according to Definition 2. Thus we delete the items above directly (Lines 2–10). And if there exist $e_{new} \prec e \wedge k(a_e) > U - N + 1 \wedge (1 - P(a_e)) \times P_{new}(e) < q$, we update the interval of e as $(k(a_e), k(e)]$ (Lines 11–15). Otherwise, we recalculate the stabbing query interval of item e by using its dominating set (Lines 16–17).

Algorithm 3. Maintaining $S_{N,q}$

```
1  begin
2      foreach compute node P_i (1 ≤ i ≤ n) do
3          if P(e_new) > (1 − q) then
4              delete ∀e ∈ {e'|e_new ≺ e' ∧ e' ∈ S_{N,q}} and their intervals from RBI;
5          else
6              compute the BD(e_new) dominated by e_new;
7              foreach e ∈ BD(e_new) do
8                  P_new(e) := P_new(e) × (1 − P(e_new));
9                  if P_new(e) < q then
10                     delete e and its interval from RBI;
11                 else
12                     P_sky(e) := P_sky(e) × (1 − P(e_new));
13                     if P_sky(e) < q ∧ k(a_e) > U − N + 1 then
14                         if P_new(e) × (1 − P(a_e)) < q then
15                             e.l := k(a_e);
16                         else
17                             recalculate e.l;
```

6 Experimental Evaluation and Analysis

6.1 Experimental Setup

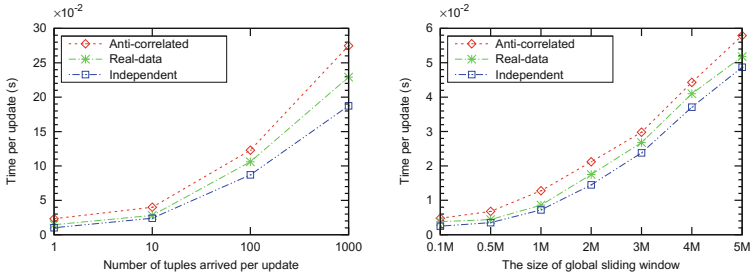
We conduct all our experiments with real deployment in the TianHe-1A HPC environment. All the algorithms are implemented in C++ running on the Linux OS. The data sources we adopted in the experiments are Real data, Independent data, and Anti-correlated data. Table 2 summarizes the parameters and their default values are indicated in bold. Note that, $1M$ in Table 2 means that there are 1×10^6 streaming items in W.

6.2 Experimental Results

Figure 2 shows the experimental results with different parameters, such as m, $|W|$, d, and q. We can know that in Fig. 2(a)–(d), the time for each update increases as the increase of sliding granularity, sliding window size, item dimensionality, and decreases as the increase of probability threshold.

Table 2. System parameters

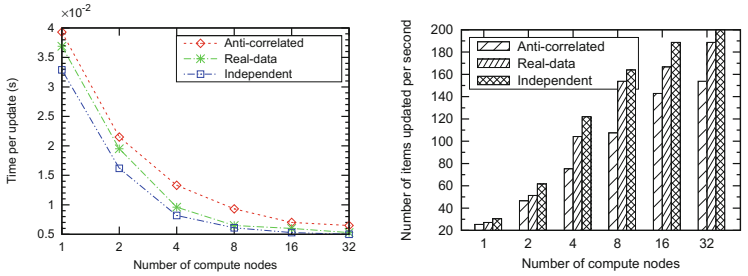
Parameter	Meaning	Values
m	Sliding granularity	$\mathbf{10^0}$ 10^1 10^2 10^3
$\|W\|$	Global window size	0.1M 0.5M **1M** 2M 3M 4M 5M
t	Number of compute nodes	1 2 **4** 8 16 32
d	Item dimensionality	2 **3** 4 5 6
q	Probability threshold	0.1 0.3 **0.5** 0.7 0.9



(a) Sliding granularity (b) Window size

(c) Dimensionality (d) Probability threshold

Fig. 2. Overall performance.



(a) Time for each update (b) Update number per second

Fig. 3. Performance versus number of computational nodes.

We can easily see that in Fig. 3, the time per update decreases gradually as the increase of t from 1 to 32. The main reason is that if the number of compute nodes is large, the items processed by each compute node will be small.

7 Conclusion and Future Work

In this paper, we define parallel n-of-N skyline queries over uncertain data streams, and propose a novel and efficient framework to answer these queries. To the best of our knowledge, this is the first work to study the problem. Extensive experiments are conducted to demonstrate our proposals are effective and high-efficiency. In addition, we will study the k-dominance skyline queries over high-dimensional uncertain data streams in future work, in order to meet the query requirements of users' special preferences.

Acknowledgement. This work was supported by the National Key Research and Development Program of China (Grant No. 2018YFB0203801), the National Natural Science Foundation of China (Grant No. 61502511, 61572510) and China National Special Fund for Public Welfare (Grant No. GYHY201306003).

References

1. Godfrey, P., Shipley, R., Gryz, J.: Algorithms and analyses for maximal vector computation. VLDB J. Int. J. Very Large Data Bases (VLDBJ) **16**(1), 5–28 (2007)
2. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: Proceedings of the International Conference on Very Large Data Bases (VLDB), pp. 15–26 (2007)
3. Yang, Z., Yang, X., Zhou, X.: Uncertain dynamic skyline queries for uncertain databases. In: Fuzzy Systems and Knowledge Discovery, pp. 1797–1802 (2015)
4. He, G., Chen, L., Zeng, C., Zheng, Q., Zhou, G.: Probabilistic skyline queries on uncertain time series. Neurocomputing **191**, 224–237 (2016)
5. De Matteis, T., Girolamo, S.D., Mencagli, G.: Continuous skyline queries on multicore architectures. Concurr. Comput.: Pract. Exp. **28**(12), 3503–3522 (2016)

6. Park, Y., Min, J.K., Shim, K.: Efficient processing of skyline queries using MapReduce. IEEE Trans. Knowl. Data Eng. **29**(5), 1031–1044 (2017)
7. Yang, Y., Wang, Y.: Efficient probabilistic skyline computation against n-of-N data stream modal. J. Softw. **23**, 550–564 (2012)
8. Zhang, W., Li, A., Cheema, M.A., Zhang, Y., Chang, L.: Probabilistic n-of-N skyline computation over uncertain data streams. World Wide Web **18**(5), 1331–1350 (2015)
9. Li, X., Wang, Y., Li, X., Wang, Y.: Parallelizing skyline queries over uncertain data streams with sliding window partitioning and grid index. Knowl. Inf. Syst. **41**(2), 277–309 (2014)
10. Li, X., Wang, Y., Li, X., Wang, Y.: Parallel skyline queries over uncertain data streams in cloud computing environments. Int. J. Web Grid Serv. **10**(1), 24–53 (2014)

A Recommender System with Advanced Time Series Medical Data Analysis for Diabetes Patients in a Telehealth Environment

Raid Lafta[1,2], Ji Zhang[1(✉)], Xiaohui Tao[1], Jerry Chun-Wei Lin[3,4(✉)], Fulong Chen[5], Yonglong Luo[5], and Xiaoyao Zheng[5]

[1] Faculty of Health, Engineering and Sciences, University of Southern Queensland, Toowoomba, Australia
Ji.Zhang@usq.edu.au
[2] Computer Center, University of Thi-Qar, Thi-Qar, Iraq
[3] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
jerrylin@ieee.org
[4] Department of Computing, Mathematics, and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
[5] School of Computer and Information, Anhui Normal University, Wuhu, China

Abstract. Intelligent technologies are enjoying growing popularity in a telehealth environment for helping improve the quality of chronic patients' lives and provide better clinical decision-making to reduce the costs and workload involved in their daily healthcare. Obtaining a short-term disease risk prediction and thereby offering medical recommendations reliably and accurately are challenging in teleheath systems. In this work, a novel medical recommender system is proposed based upon time series data analysis for diabetes patients. It uses three decomposition methods, i.e., dual-tree complex wavelet transform (DTCWT), fast Fourier transformation (FFT) and dual-tree complex wavelet transform-coupled fast Fourier transform (DWCWT-FFT), with least square-support vector machine (LS-SVM) for short-term disease risk prediction for diabetes disease patients which then generates appropriate recommendations on their need to take a medical test or not on the coming day based on the analysis of their medical data. A real-life time series dataset is used for experimental evaluation. The experimental results show that the proposed system yields very good recommendation accuracy and can effectively reduce the workload for diabetes disease patients in conducting daily body tests.

Keywords: Decomposition methods · Recommender system
Diabetes disease patients · Time series prediction · Telehealth · SVM

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 185–192, 2018.
https://doi.org/10.1007/978-3-319-98812-2_15

1 Introduction

According to World Health Organization (WTO), chronic diseases are causing the death for 50% of people worldwide in recent years [1], and they require more and more medical attentions and resources in today's increasingly aged societies. Diabetes, one of the most common chronic diseases, is a major health problem in the world and the rates of its incidence are significantly rising [10].

Telehealth systems serve as real time and convenient platforms for healthcare practitioners and chronic diseases patients to exchange information easily in consultation, diagnosis and treatment [2], and consequently have enjoined fast developments in many countries in recent years due to fast service delivery and its low operational cost. Due to the importance of disease risk prediction on the patients' life who suffering from the chronic diseases [8] such as diabetes as well as the urgency of improving the analytic techniques used for this regard, great efforts are needed to enhance the quality of evidence-based decisions and recommendations in a telehealth environment. Diabetes disease patients often need to undertake various daily medical tests in order to monitor their overall health conditions through the telehealth system. However, in the current practice, carrying out various medical tests by diabetes disease patients every day may bring lots of inconvenience and even burden, and thus affects their overall life quality.

Generating accurate recommendations is an essential function in telehelth systems, which is often based on the prediction of patients' short-term disease risk. In literature, the assessment and prediction of various diseases have been studied by using data mining techniques and statistical tools for different healthcare and medical issues [3,4]. Although most of these studies have been achieved a reasonable level of predictive accuracy, most of them focused on the long-term medical prediction instead of short-term prediction which is studied in our work.

The major scientific contributions and features of our system are summarized as follows.

- The system utilizes three decomposition methods, including DTCWT, FFT and DWCWT-FFT;
- The statistical features extracted from these methods are then separately input into the Least Square-Support Vector Machine classifier (LS-SVM) to predict the necessity of taking body test on the next day in advance;
- We use a majority vote based ensemble technique to combine the prediction results based on the three individual decomposition methods for producing the final recommendation for diabetes patients;
- We compare our system with the existing work conducted to tackle the exactly same issue to establish the superiority of our technique.

2 Proposed Recommender System

2.1 An Overview of Our System

Figure 1 illustrates the overall architecture of our recommender system used for diabetes patients in the telehealth environment. First, the time series medical

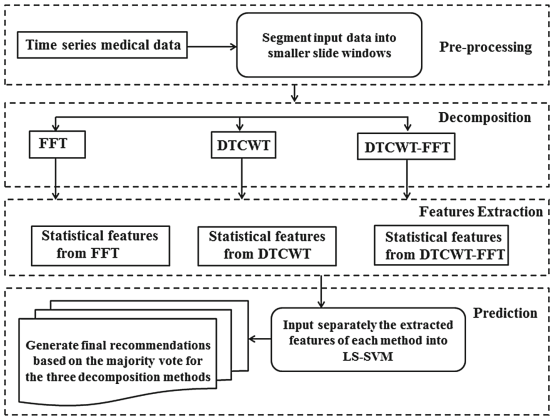


Fig. 1. The architecture of our recommendation system

data of a given patient is pre-processed, which is performed off-line, by segmenting them into smaller overlapped sliding windows based on the size of the sliding window used in the data analysis. Then, the three decomposition methods – DTCWT, FFT and DTCWT-FFT – are separately applied to decompose the segmented time series data of patients. The LS-SVM is used with each decomposition method to test its ability to classify the patient's condition. The final recommendation is then taken based on the ensemble mechanism using the majority vote approach for the three decomposition methods in order to produce a binary accurate recommendation concerning whether the patient needs to take a medical test on the coming day or not.

2.2 Dual Tree Complex Wavelet Transformation

The drawbacks of DWT are ameliorated by using the Dual Tree Complex Wavelet Transformation (DTCWT) which offers a better time-frequency representation of signals [5]. It is an improved version of wavelet transformation that is designed to tackle some limitations in the discrete wavelet transform.

In our system, DTCWT is adopted to decompose the input time series data into sub-bands of *delta, theta, alpha, beta* and *gamma*. Each DTCWT coefficient has two parts real and imaginary. As a result, ten sub-bands in total obtained after four-level decomposition (five sub-bands for each part).

From each frequency sub-band, six different statistical features can be extracted. The extracted features are mean of coefficients of the absolute values, average power of the coefficients, standard deviation of the coefficients, ratio of the absolute mean values of coefficients of adjacent sub-bands, kurtosis of the coefficients and skewness of the coefficients respectively.

2.3 Fast Fourier Transformation

The Fast Fourier Transformation (FFT) is one of the most efficient techniques used to compute the Discrete Fourier Transformation (DFT) and its inverse. In many studies, it is used as a windowing technique like the wavelet transformation [6,9].

For each sliding window, five frequency bands (i.e., *alpha*, *beta*, *gamma*, *delta*, and *theta*) are obtained using the fast Fourier transformation.

From each frequency band, eight different statistical features can be extracted. The extracted features are denoted by X_{Min}, X_{Max}, X_{SD}, X_{Med}, X_{Mean}, X_{RG} (for Range), X_{FQ} (for the First Quartile) and X_{SQ} (for the Second Quartile), respectively. The best performing features are dataset dependent. Some series data are symmetrically distributed while others may have a more skewed distribution.

In our work, the extracted features from each frequency band are grouped into one vector and used as the input to the LS-SVM to predict the patient's condition.

2.4 Hybrid Method of Dual Tree Complex Wavelet Transform with Fast Fourier Transform (DTCWT-FFT)

In this method, we apply 1D dual-tree complex wavelet transform to the input time series data of patients for four-level DTCWT decomposition, and then applies the fast Fourier transform to each DTCWT sub-bands and takes the magnitude of these coefficients. In this way, the levels of the Fourier spectrum vectors are used as a features set and the LS-SVM is used to classify the input time series data into one of two classes: test required or no test required.

The proposed method for generating short-term medical recommendation can be summarized as follows:

1. Get the input time series data of patients $x(n)$ where $n \in [1, N]$.
2. Apply four-levels DTCWT decomposition to the input time series data. Let the output time series be y{1}, y{2}, y{3}, y{4} and z{4} for levels 1, 2, 3, and 4 respectively.
3. Apply forward FFT to y{1}, y{2}, y{3}, y{4} and z{4} and then take the logarithm of the Fourier spectrum. Let the generated features be F{1}, F{2}, F{3}, F{4}, and F{5} respectively.
4. All the generated features vectors enter to the LS-SVM classifier to classify the input time series data of patient.

3 Experimental Results

3.1 Diabetes Dataset

The diabetes dataset obtained from the Repository of Machine Learning Databases by Washington University [7]. The collected data contain measurements taken multiple times per day from 70 patients. Blood glucose measurements, symptoms and insulin treatments were recorded with timestamps for each patient, over the course of several weeks to months.

Each record in the diabetes dataset consists of four fields about the date of measurement, time of measurement, the code of measurement and the value of measurement.

For the purpose of evaluating our system, the dataset is divided into two parts: the training set and the testing set. The three transfer methods are trained using the training set and then validated using the testing set as the ground truth result. In our study, 75% of the dataset was partitioned as the training data while the remaining 25% was used as testing data.

3.2 Performance Evaluation Measurements

To evaluate the performance of the proposed system, we have proposed three performance metrics for this work, namely *accuracy*, *workload saving* and *risk*. Accuracy refers to the percentage of correctly recommended days against the total number of days for which recommendations are provided. Workload saving refers to the percentage of the total number of days when recommendations are provided for skipping the medical test against the total number of days in the training set. Risk refers to the percentage of the days with risky recommendation against the total number of days in the training set.

3.3 Recommendation Effectiveness of Our System

Recommendation Effectiveness of Our System Using FFT. Based on our previous work [8,9], it was found that the obtained prediction results of our system were not good enough when the features were not appropriately selected from a time series data and vice versa. Thus, the statistical features of FFT were tested separately to evaluate the prediction accuracy of the proposed system.

Furthermore, the patient discrimination ability of the eight statistical features $\{X_{Min}, X_{Max}, X_{SD}, X_{Med}, X_{Mean}, X_{RG}, X_{FQ}, X_{SQ}\}$ is performed using t-test. The p-values of the eight statistical features of five waveforms for two different classes including taking a test or not needed using t-test are presented in Table 1. It can be seen that the last four waves (*theta, alpha, beta and gamma*) with the statistical features of rang, mean, median, standard deviation, max, and min provide a significantly difference ($P < 0.003$). Thus, the six features of $\{X_{Min}, X_{Max}, X_{SD}, X_{Med}, X_{Mean}, X_{RG}\}$ of four waves are extracted from FFT sub-bands were used to evaluate the performance of our system for predicting the patient's condition one day in advance. The vector of features is then entered into LS-SVM classifier to decide whether a given patient needs to take a medical measurement on day in advance or not.

Based on the results in Table 3, it can be seen that when using the six statistical features with the four waves, our system can achieve an accuracy over 90%, a workload saving over 63% while the risk is lower than 5 %, indicating that our recommendation system is highly accurate and able to significantly reduce the workload for chronic diabetes disease patients to take up their daily medical tests with a low health risk.

Recommendation Effectiveness of Our System Using DTCWT. In this experiment, the extracted statistical features from the high-frequency sub-bands (i.e., y1, y2, and y3) were also tested separately to evaluate the prediction accuracy of the proposed system.

To achieve the best possible performance of our system, the statistical features extracted from the high-frequency sub-bands (i.e., y1, y2, and y3) are quantified using t-test. Table 2 shows the P-values of the six features extracted from five waveforms of need or not to take a medical test. On the basis of these results, we can be clearly observed that the five statistical features of average power, ratio of mean values, standard deviation, skewness and kurtosis extracted from the sub-bands (*alpha, beta and gamma*) are provide a highly deference ($P < 0.001$). Hence, the five statistical features with the three sub-bands DTCWT are selected to present the time series data of patient.

Our findings showed that combining all the five statistical features for the three high-frequency sub-bands yielded a high prediction accuracy with an average accuracy of 91%, workload saving 63% and risk 4%. The obtained results showed that the statistical features were able to reveal the characteristics of time series data of patients, and to identify patient's condition for short-time disease risk prediction.

Table 1. p-values of statistical features for all waveforms

Waveforms	p-values of X_{FQ}	p-values of X_{SQ}	p-values of X_{SD}	p-values of X_{MAX}	p-values of X_{RG}	p-values of X_{Min}	p-values of X_{Med}	p-values of X_{Mean}
Delta	0.0425	0.0215	6.5214E−09	3.1245E−04	0.0695	0.0425	0.0525	4.2586E−11
Theta	0.0612	0.0325	2.3125E−06	6.2548E−07	0	3.5271E−05	4.2154E−06	0
Alpha	0.4325	0.0345	0	0	5.6243E−05	0	0	0
Beta	0.2154	0.6258	0	5.3641E−09	0	8.2546E−10	0	2.2547E−11
Gamma	0.9457	0.5054	8.9554E−07	0	0	0	0	0

Table 2. p-values of statistical features extracted form the five waveforms of DTCWT

Waveforms	p-values of mean	p-values of average power	p-values of ratio of mean values	p-values of standard deviation	p-values of skewness	p-values of kurtosis
Delta	0.0914	0.0754	0.0541	8.7894E−06	0.0465	0.0254
Theta	0.0456	0.1245	0.3254	0.0312	0.2584	0.4512
Alpha	0.1524	0	0	0	0	5.2364E−10
Beta	0.6324	0	5.8254E−09	0	0	0
Gamma	0.5214	7.5417E−05	0	6.2548E−07	7.5588E−09	0

Recommendation Effectiveness of Our System Using a Hybrid Transform Method (DTCWT-FFT). In order to improve this method, the present study applies the a dual-tree complex wavelet to the input time series data of patients for four-level DTCWT decomposition, and then applies the fast Fourier

transform to each DTCWT sub-bands and takes the magnitude of these coefficients. The extracted features are entered to the LS-SVM classifier to decide whether the patient who is suffering from chronic diabetes disease needs to take a medical body test one day in advance or not.

The proposed system using a hybrid method yields an improved performance compared with the previous two methods. Our system using a hybrid method able to improve the accuracy performance from 90% to 93% while there is no significant difference in the value of workload saving where the workload saving rate is a very close to the two previous methods. The recommendation risk of our system is also lower than the two competitive approaches.

Recommendation Effectiveness Based on the Majority Vote of the Three Decomposition Methods. First, we apply the three methods to process the time series medical data to facilitate the subsequent data analytic. Then, the statistical features extracted from each decomposition method are separately entering into the least square-support vector machine classifier to predict the necessity of taking body test. The final recommendation of a given medical measurement is considered based on applying the majority vote of the three decomposition methods to decide whether the patient who is suffering from chronic diabetes disease needs to take a medical body test one day in advance or not.

Based on Table 3, the proposed method based on majority vote technique had the ability to classify the patients' condition with a high accuracy over 96%, while the risk is lower than 1.5%.

Table 3. The averaged performance of the three decomposition methods and the proposed method

Method used	Accuracy (%)	Saving (%)	Risk (%)
FFT	90.90	63.40	04.75
DTCWT	91.00	63.20	04.30
DTCWT-FFT	93.60	64.00	03.20
Proposed method	**96.00**	**64.50**	**01.50**

4 Conclusions and Future Research Directions

In this work, we propose a recommendation system supported by three decomposition methods including dual-tree complex wavelet transform, fast Fourier transform and dual-tree complex wavelet-coupled fast Fourier transform– to provide the patients suffering from chronic diabetes disease with appropriate

recommendations in a telehealth environment. This study applies three decomposition methods which effectively analyze the medical time series data and input separately the extracted statistical features from each method to the LS-SVM to generate the accurate, reliable recommendations for chronic diabetes disease patients. The final recommendation is taken according to the majority vote of the three decomposition methods.

In future, we will apply other ensemble techniques, such as Adaboost and boosting, to generate recommendations and conducting a comparative study on those different ensemble models.

Acknowledgment. This research was partially supported by Guangxi Key Laboratory of Trusted Software (No. kx201615), Shenzhen Technical Project (JCYJ20170307151733005 and KQJSCX20170726103424709), the general research project of National Science Foundation of China (No. 61572036, No. 61672039, No. 61772034) and Anhui Provincial Natural Science Foundation (1808085MF172).

References

1. Kuh, D., Shlomo, Y.B.: A Life Course Approach to Chronic Disease Epidemiology. Oxford University Press, Oxford (2004)
2. Dewar, A.R., Bull, T.P., Malvey, D.M., Szalma, J.L.: Developing a measure of engagement with telehealth systems: the mHealth technology engagement index. J. Telemed. Telecare **23**, 248–255 (2017)
3. Mohktar, M.S., et al.: Predicting the risk of exacerbation in patients with chronic obstructive pulmonary disease using home telehealth measurement data. Artif. Intell. Med. **63**(1), 51–59 (2015)
4. Krishnaiah, V., Narsimha, D.G., Chandra, D.N.S.: Diagnosis of lung cancer prediction system using data mining classification techniques. Int. J. Comput. Sci. Inf. Technol. **4**(1), 39–45 (2013)
5. Das, A.B., Bhuiyan, M.I.H., Alam, S.S.: Classification of EEG signals using normal inverse Gaussian parameters in the dual-tree complex wavelet transform domain for seizure detection. Signal Image Video Process. **10**(2), 259–266 (2016)
6. Deo, R.C., Wen, X., Qi, F.: A wavelet-coupled support vector machine model for forecasting global incident solar radiation using limited meteorological dataset. Appl. Energy **168**, 568–593 (2016)
7. AIM-94 data set provided by Michael, K., MD. Ph.D. Washington University, St. Louis, MO, US. https://archive.ics.uci.edu/ml/datasets/diabetes
8. Lafta, R., et al.: An intelligent recommender system based on predictive analysis in telehealthcare environment. Web Intell. **14**(4), 325–336 (2016). IOS press
9. Lafta, R., et al.: A fast fourier transform-coupled machine learning-based ensemble model for disease risk prediction using a real-life dataset. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 654–670. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57454-7_51
10. Temurtas, H., Yumusak, N., Temurtas, F.: A comparative study on diabetes disease diagnosis using neural networks. Expert Syst. Appl. **36**(4), 8610–8615 (2009)

Information Networks and Algorithms

Edit Distance Based Similarity Search of Heterogeneous Information Networks

Jianhua Lu[1(✉)], Ningyun Lu[2], Sipei Ma[1], and Baili Zhang[1]

[1] School of Computer Science and Engineering, Southeast University, Nanjing, China
{lujianhua,masp,zhangbl}@seu.edu.cn
[2] College of Automation Engineering,
Nanjing University of Aeronautics and Astronautics, Nanjing, China
luningyun@nuaa.edu.cn

Abstract. In this big data age, extensive requirements emerge in data management and data analysis fields. Heterogeneous information networks (HIN) are widely used as data models due to their rich semantics in expressing complex data correlations. The data similarities other than the exact matches are required in many data mining, data analysis and machine learning algorithms. Graph edit distance (GED) is one of the feasible methods on HIN similarity measuring. In this paper, we firstly extend the concept of GED in homogeneous graphs to the heterogeneous information networks by introducing newly defined edit operations. The metapath-based approximation method is then proposed to improve the performance of full database similarity search, in which a upper bound and a lower bound, both of polynomial time complexity, are utilized as filters. Finally, comprehensive experimental results show the proposed method outperforms the existed method in terms of computational efficiency, bound tightness and similarity filtering capability.

Keywords: Heterogeneous information network · Similarity search
Graph edit distance · Metapath · Lower bound · Upper bound

1 Introduction

In big data age, extensive requirements emerge in data management and analysis fields. A suitable data model is essential for the effectiveness and efficiency. Heterogeneous information networks (HIN) are widely used as data models due to its rich semantics in expressing complex data correlations [1]. Different from the traditional data graphs, or homogeneous information networks, in which the entities and relationships are of same types, the entities (vertices) in heterogeneous information networks are multi-typed and there are diverse relationships (edges) among the entities.

In some data applications, it is important to measure the data similarities other than to acquire the exact matches. As one of the most widely used and feasible methods, Graph Edit Distance (GED) is proposed to calculate the similarity of homogeneous graphs [2]. Unfortunately, the cost of GED computation

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 195–202, 2018.
https://doi.org/10.1007/978-3-319-98812-2_16

is exponential to the number of graph vertices, which is too expensive to be applicable in large scale graphs. Therefore, feature-based GED approximation techniques are proposed to improve graph search performance. Heuristic methods are proposed to obtain upper and lower bounds of GED to improve the efficiency of graph similarity searches [3,4].

Although the heterogeneous information networks share the same basic notions as the homogeneous graphs, i.e. they are both composed of vertices and edges. There are issues need to be addressed in similarity measuring. Furthermore, there are more options in performance improving as HINs have richer information that can be utilized.

In this paper we extend the concept of Graph Edit Distance in homogeneous graphs to the heterogeneous information networks by introducing the newly defined edit operations. A star-based approach is firstly proposed by revising the existed work on homogeneous graphs. Due to the unique characteristics of heterogeneous information networks [5], metapaths are then utilized as basic semantic components and the metapath mapping distance is proposed to approximate the HIN Edit Distance. Additionally, metapath mapping distance is used to support graph similarity search by obtaining upper and lower bounds of HIN edit distance in polynomial time. In summary, the contributions of this paper are as follows.

1. Graph edit distance is extended from homogeneous graphs into heterogeneous information networks, noted as HIN-ED, in which the vertex types are taken into account.
2. Metapaths are utilized as the features of HINs, and the metapath mapping distance is proposed to approximate HIN-EDs.
3. The metapath-based lower bound and upper bound, both of polynomial time complexity, are introduced to assist the similarity search.
4. Comprehensive experimental studies are conducted to evaluate the efficiency, bound tightness, scalability and filtering performance of the proposed algorithms.

2 HIN Edit Distance and Similarity Search

Many concepts, such as *information network*, *heterogeneous information network(HIN)*, *network schema* and *meta path*, were proposed by Han and Sun [7].

The existed edit distance metric is proposed to measure the similarities of homogeneous graphs. We extend it by adding new edit operations to support similarity search of heterogeneous information networks. The edit operations are conducted on HIN with no self-loops, multi-edges or edge labels.

A heterogeneous information network can be transformed to another one by performing a sequence of edit operations. We consider six edit operations.

1. Vertex Insertion. Add a vertex into the network;
2. Edge Insertion. Add an edge into the network;
3. Vertex Deletion. Remove a vertex from the network;

4. Edge Deletion. Remove an edge from the network;
5. Vertex Relabeling. Change the label of one vertex into another one;
6. Vertex Type Substitution. Change the type of one vertex into another one.

Vertex labels are subordinate to vertex types. The cost of vertex type substitution is 2 since it will cause the vertexs label being changed.

Definition 1. *Heterogeneous Information Network Edit Distance (HIN-ED). Given heterogeneous information networks g_1 and g_2, the edit distance of g_1 and g_2, denoted as $\lambda(g_1,g_2)$, is the number of edit operations in the optimal edit operation alignment that make g_1 reach g_2.*

Let $P=(p_1,p_2,\cdots,p_k)$ be an edit operation alignment transforming g_1 to g_2. Accordingly, there is a sequence of information networks $g_1=h_0\rightarrow h_1 \rightarrow\cdots\rightarrow h_k=g_2$, where $h_{i-1}\rightarrow h_i(1\leq i\leq k)$ indicates that h_i is derived from performing p_i over h_{i-1}.

The purpose of full similarity search is to find all the heterogeneous information networks in database that is similar to a given query network. A distance threshold is commonly used to measure the similarity by testing whether the distances between networks are qualified or not.

Algorithm 1. HIN-Sim: Full Similarity Search of Heterogeneous Information Networks.

Input: A query graph q and a graph database D; Distance threshold ω;
Output: All graphs g in D where $\lambda(g, q) \leq \omega$;

```
1: for each graph g∈D do
2:     if L_b(g,q) ≥ ω then
3:         continue;
4:     end if
5:     if U_b(g,q) ≤ ω then
6:         report g as a result;
7:         continue;
8:     end if
9:     if λ(g,q) ≤ ω then
10:        report g as a result;
11:    end if
12: end for
```

Algorithm 1 illustrates the process of full similarity search. Given a query network q and a distance threshold ω, it is inefficient to calculate and test the distances of q and every network g in D. Therefore, a lower bound and an upper bound are used as filters. The lower bound, $L_b(g,q)$, is first calculated and tested against to filter out the unqualified data (line 2–4). Then, the upper bound,$U_b(g,q)$, is calculated and tested to get the definitely qualified data (line 5–7). Finally, the edit distance, λ (g, q), is calculated and tested on the data that are in between (lines 9–11).

It is obvious that the performance of Algorithm 1 depends on the bounding tightness and the computing efficiency of the bounds. The closer the bounds to the HIN-EDs are, the better the filtering capability is, and the less the HIN-EDs are calculated.

3 Metapath-Based HIN-ED Lower and Upper Bounds

Zeng and Tung proposed to bound the graph edit distance based on stars since the star structure is one of the basic structures of graph [4]. While, metapath is a special structure depicting the composite relationships of heterogeneous information networks [5,7,8]. We use metapaths as the basic components of HINs to bound the HIN edit distance.

Definition 2. *Metapath Schema Coverage. A HIN schema T_G=(V_s, E_s, T_s, α_s) is covered by a metapath set P_m if for each e∈E_s there is a metapath p∈P_m that contains e.*

Definition 3. *Metapath HIN Coverage. A HIN G is covered by a metapath instance set P if (1) the schema of G is covered by metapath set P_m; (2) any metapath instance in P is an instance of some metapath in P_m; (3) all edges of G appears in some metapath instance in P. We call P is the metapath representation of HIN g, denoted as P(G).*

Proposition 1. *Metapath Edit Distance. The edit distance of two metapath instances p_1 and p_2 is:*

$$\lambda(p_1, p_2) = \sum_{i=1}^{n} M(p_1.N_i, p_2.N_i)$$
$$T(r_1, r_2) = \begin{cases} 2 & if \alpha(v_1) \neq \alpha(v_2) \\ 1 & if \alpha(v_1) = \alpha(v_2) \, and \, \beta(v_1) \neq \beta(v_2) \\ 0 & if \alpha(v_1) = \alpha(v_2) \, and \, \beta(v_1) = \beta(v_2) \end{cases} \quad (1)$$

p.N_i is the i^{th} node in path p.

Proof. The proof is simple and omitted.

Definition 4. *Given two metapath instance multisets P_1 and P_2 with identical cardinalities, The mapping distance from P_1 to P_2, denoted as B:$P_1 \rightarrow P_2$, is a bijection and defined as follows.*

$$\zeta(S_1, S_2) = min_B \sum_{p_i \in P_1} \lambda(p_i, B(p_i)) \quad (2)$$

Computing ζ(P_1,P_2) is similar to computing ζ(S_1,S_2) introduced in [4].

Definition 5. *Metapath Mapping Distance. The metapath mapping distance $\mu(g_1,g_2)$ of HIN g_1 and g_2 is defined as:*

$$\mu(g_1, g_2) = \zeta(P_1, P_2) \quad (3)$$

We will analyze the relationship between the metapath edit distance and each edit operations in GED sequence.

Proposition 2. *Given two HINs g_1 and g_2, the metapath mapping distance $\mu(g_1,g_2)$ of g_1 and g_2 satisfies the following inequality:*

$$\begin{aligned}\mu(g_1,g_2) \leq & max\{max\{\delta(g_1)^{L-2},\delta(g_2)^{L-2}\}\times L,\\ & max\{(\delta(g_1)^{2M})^{L-2},(\delta(g_2)^{2M})^{L-2}\}\times 2\}\\ & \times\lambda(g_1,g_2)\end{aligned} \tag{4}$$

where L is the length of metapath, and the maximum number of singular-typed neighbours of all the vertices in g is noted as δ(g).

Proof. The proof is omitted.

Similar to the star mapping distance, certain edit operations may also be redundantly calculated in metapath mapping distance. For example, one vertex relabeling operation on v_0 is conducted to edit g_1 to g_2. Assume that label of v_0 is changed from σ_1 to σ_2, and σ_1 doesn't exist in g_2. It is clear that metapath mapping distance counts the number of metapath instances containing v_0 and adds it up, of which only 1 unit cost contributes to HIN-ED.

Given a HIN G=(V, E, T_v, α, L, β) and a coverage metapath instance set P, the metapath degree of a vertex type t, noted as pathDeg(t), is the minimum number of edges incident with the vertices of type t that are contained by P.

Proposition 3. *Metapath-based Upper Bound. Given the metapath mapping distance of HINs g_1 and g_2, $\mu(g_1,g_2)$, the metapath-based upper bound of $\lambda(g_1,g_2)$, denoted as $\tau(g_1,g_2)$, satisfies the following equation:*

$$\begin{aligned}\tau(g_1,g_2)=\mu(g_1,g_2)-M(g_1,g_2)\geq\lambda(g_1,g_2)\\ M(g_1,g_2)=\sum_{i=1}^{n}(min(pathDeg(g_1.a(V_i)),pathDeg(g_2.a(V_i)))-1)\\ \times(min(|g_1.a(V_i)|,|g_2.a(V_i)|)-|g_1.a(V_i)\cap g_2.a(V_i)|)\end{aligned} \tag{5}$$

$g.a(V_i)$ is the set of vertices with the same type as V_i in network g.

Proof. The proof is omitted.

4 Performance Evaluation

All the experiments were conducted on a 2.60 GHz Inter(R) Core(R) i7 PC with 8 GB main memory, running Windows 7. All the algorithms were implemented in C++ and compiled by MS VC10. The dataset is generated by an revised version of the generator kindly provided by Kuramochi [6].

As the base-lines, we revised the algorithm in [4] and implemented the star-based lower bound and star-based upper bound on HIN.

Experiment 1 [Computing efficiency]. The average runtime of computing star-based lower bound ω, star-based upper bound θ, metapath-based lower bound γ and metapath-based upper bound τ were tested in this experiment. We compared the four bounds with exact HIN-ED λ on the synthetic dataset with different network sizes.

In Fig. 1, the X-axis shows the network sizes being tested, and the Y-axis shows the corresponding average runtimes of the four bounds and HIN-ED. The lengths of metapaths were set to 3 for simulating the network schema of DBLP. The cost of HIN-ED computation was exponential to the graph size and does not apply in large-scale HIN similarity measuring. For HINs, one vertex may act as a root or as a leave in a star, while its position in a metapath is fixed. So, metapath mapping is relatively simpler than star mapping. As a result, the computations of metapath-based bounds, τ and γ, outperformed that of star-based bounds, θ and ω, by 5 to 10 times generally. Since the upper bounds shrink from the mapping distances, the costs of computing upper bounds were slightly larger than that of lower bounds in both sub-structures.

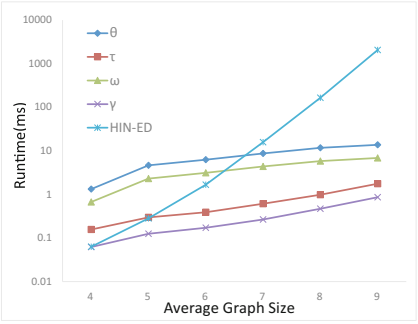


Fig. 1. Computational efficiency

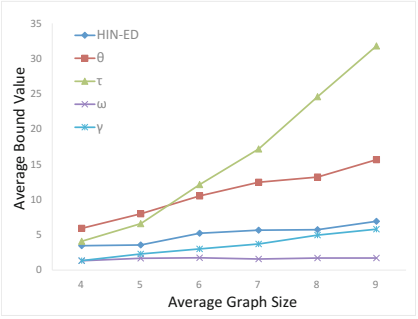


Fig. 2. Average bound value

Experiment 2 [Tightness of bounds]. In this experiment, we looked into the tightness of lower and upper bounds. The lengths of metapaths were set to 3 for computing the metapath-based bounds. Figure 2 depicts the average values of ω, γ, θ, τ and λ on the networks of different sizes.

Metapaths capture more semantics than stars, and the redundancy in metapath mapping distance is less than that in star mapping distance. So, in Fig. 2 the metapath-based lower bound, γ, was closer to λ than the star-based lower bound, ω. When the networks got bigger, ω went further from λ than γ since the increasing rate of number of stars is much faster than that of number of metapaths. As for the upper bounds, the metapath-based upper bound τ bounded more tightly than star-based upper bound θ when the sizes were relatively small, while the situation changed when the sizes got bigger.

We adopt the well known approximation ratio e=max{d/λ,λ/d} to measure the bounding tightness, where d represents the lower bounds or upper bounds. The closer to 1 e is, the tighter the bound is.

As shown in Fig. 3, the approximation ratio of the metapath-based lower bound γ gradually approached to 1 along with the increasing of the network sizes. On the other hand, the star-based lower bound ω gradually got looser from HIN-ED. To be quantitative, the average e of ω was 92.02% larger than that of γ. For upper bounds in Fig. 4, the approximation ratio of star-based upper bound tended to be stable, while the approximation ratio of metapath-based upper bound τ increased fast when the network scale was relatively bigger. The average e of θ was 26.27% tighter than that of τ.

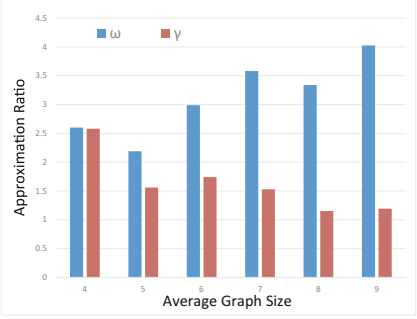


Fig. 3. Lower bound tightness

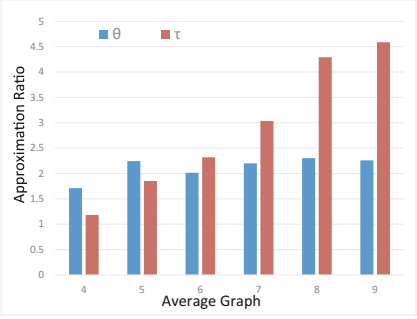


Fig. 4. Upper bound tightness

Experiment 3 [Full similarity search]. In experiment 3, we investigated the performance of similarity search by setting different upper and lower bounds as filters. The number of expensive HIN-ED computations (noted as #λ) was set as the filter criteria. Since λ cannot be solved in a polynomial time, the fewer λ is calculated, the better the filter is. Four versions of algorithm HIN-Sim were tested, HIN-Sim:ω (with star-based lower bound), HIN-Sim:γ (with metapath-based lower bound), HIN-Sim:ω+θ (with star-based lower bound and upper bound), and HIN-Sim:γ+τ (with metapath-based lower bound and upper bound).

Figure 5 gives the filter abilities of the proposed bounds. The metapath-based filters were better than the star-based filters in general, which agreed to the conclusion of the bounding tightness experiments. The upper bounds had little impact when the thresholds were small, while along with the threshold increasing, the upper bounds introduced significant impacts. Due to the parameters settings, such as the vertices of the same type may have more label options, the metapath-based upper bound performed much better than the star-based upper bound when the thresholds was larger.

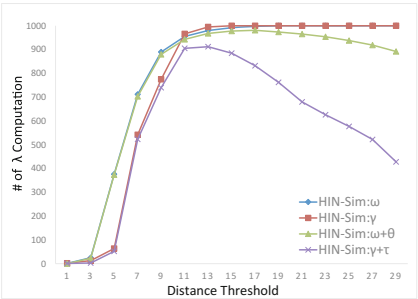


Fig. 5. Comparison of similarity search strategies

5 Conclusions

The edit distance based similarity search of heterogeneous information networks was investigated in this paper. The graph edit distance in homogeneous graphs was extended first. We utilized the metapaths to approximate HIN-ED by introducing the metapath mapping distance. The metapath-based lower bound and upper bound were derived as the filters of full similarity search. There are fewer redundant edit operations in the metapath mapping distance than that in the star mapping distance. Since metapath is a specific structure designated to capture the composite relationship semantics of HINs, the bounding effectiveness of the metapath-based lower bound was much better than the star-based lower bound. The experimental results showed the proposed methods were effective and efficient in terms of computational efficiency, bound tightness, and similarity filtering capability.

References

1. Ghosh, R., Lerman, K.: Structure of Heterogeneous Networks. arXiv **9**(6) (2009)
2. Justice, D., Hero, A.: A binary linear programming formulation of the graph edit distance. In: IEEE Transactions on PAMI, vol. 28, no. 8, pp. 1200–1214 (2006)
3. Singh, A.K.: Closure-tree: an index structure for graph queries. In: ICDE 2006, pp. 38–47 (2006)
4. Zeng, Z., Tung, A.K., Wang, J., Feng, J., Zhou, L.: Comparing stars: on approximating graph edit distance. Proc. VLDB Endow. **2**(1), 25 (2009)
5. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: PathSim: meta path based top-k similarity search in heterogeneous information networks. In: VLDB 2011, pp. 992–1003 (2011)
6. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: IEEE Transactions on ICDM 2001, p. 313 (2001)
7. Sun, Y., Han, J.: Mining heterogeneous information networks: a structural analysis approach. In: SIGKDD 2012, vol. 14, pp. 20–28 (2012)
8. Shi, C., Li, Y.: A survey of heterogeneous information network analysis. J. Latex Cl. Files **14**, 17–37 (2015)

An Approximate Nearest Neighbor Search Algorithm Using Distance-Based Hashing

Yuri Itotani, Shin'ichi Wakabayashi(✉), Shinobu Nagayama, and Masato Inagi

Graduate School of Information Sciences, Hiroshima City University,
3-4-1, Ozukahigashi, Asaminami-ku, Hiroshima 731-3194, Japan
yuri@lcs.info.hiroshima-cu.ac.jp,
{wakaba,s_naga,inagi}@hiroshima-cu.ac.jp

Abstract. This paper proposes an approximate nearest neighbor search algorithm for high-dimensional data. The proposed algorithm is based on a distance-based hashing called adaptive flexible distance-based hashing (AFDH). For a given query, AFDH returns a small-sized candidate set of nearest neighbors, and the one closest to the query is selected as the final result. The main advantage of the proposed algorithm is that, without fine tuning of parameter values of the algorithm, good search results can be obtained. Experimental results show that the proposed algorithm produces satisfactory results in terms of quality of results as well as execution time.

Keywords: Nearest neighbor search
Adaptive flexible distance-based hashing

1 Introduction

The nearest neighbor search problem is a well-known and fundamental problem in many applications such as machine learning, pattern recognition, computer graphics, information retrieval, and so on [2,4,7]. The problem is to find a data point in a set of data points D, which is closest to a given query q with respect to a defined distance measure. A typical and important version of the nearest neighbor search problem is the case that the distance measure is the ordinary Euclidean distance, and the dimension of the data space is very high such as $d \geq 20$. In this paper, this type of the nearest neighbor search problem is discussed.

To shorten the computation time of nearest neighbor search, various approaches have been proposed [1,3,5,6]. Most of existing algorithms for this problem can be classified into two categories: search algorithms using tree-based data structures and approximate nearest neighbor search algorithms based on hashing. Typical examples of tree-based data structures used in nearest neighbor search algorithms are KD-tree, R-tree and its variants, B+-tree, and Cover

This work was supported by JSPS KAKENHI Grant Number 17K00188.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 203–213, 2018.
https://doi.org/10.1007/978-3-319-98812-2_17

tree [6,7]. Although most of search algorithms using tree-based data structures can produce the exact result, they may not often be very efficient for the high-dimensional and large scale dataset.

Approximate nearest neighbor search algorithms, on the other hand, can improve the efficiency of search in computation time and space at the sacrifice of exactness of solutions. There are several types of approximate nearest neighbor search algorithms. Among them, algorithms using hashing are well-known. Various kinds of hashing techniques, such as locality-sensitive hashing (LSH), spectral hashing, etc., have been proposed to find efficiently approximate nearest neighbors with good accuracy [1,3].

In this paper, we propose an approximate nearest neighbor search algorithm for high-dimensional data based on hashing. The hashing technique adopted in the proposed algorithm is an extension of flexible distance-based hashing (FDH) [8]. For a given query, FDH returns a small-sized candidate set of nearest neighbors. Among them, the one closest to the query is selected as as the final result.

The original FDH has a drawback that there always exist some regions (i.e., subspaces of the entire data space) which would not be selected as candidate regions having nearest neighbors. To overcome this drawback of FDH, we newly introduce two ideas into FDH. One is the *furthest δ-neighbor*, and the other is the adaptive control of searching regions. Introducing those two ideas into FDH, we newly propose an extension of FDH, called *adaptive flexible-distance based hashing* (AFDH, for short). Experimental results show the effectiveness and efficiency of the proposed approximate nearest neighbor search algorithm based on AFDH.

The rest of the paper is organized as follows. Section 2 describes the nearest neighbor search problem and flexible distance-based hashing (FDH). Section 3 introduces an extension of FDH called AFDH, and proposes a new approximate nearest neighbor search algorithm based on AFDH. Section 4 shows experimental evaluation of the proposed AFDH-based nearest neighbor search algorithm. Finally, Sect. 5 concludes the paper.

2 Preliminaries

2.1 Nearest Neighbor Search

The nearest neighbor search problem is formally defined as follows. Assume that a dataset $D = \{p_1, \ldots, p_n\}$ with n elements is given. Each element in D consists of d attributes $(r_1, \ldots, r_d)$, and each attribute r_i is a real number within the range of $[L_i, U_i]$, where L_i and U_i are the lower and upper bounds of attribute r_i, respectively. From the definition, each element in D can be considered as a data point in d-dimensional space $\boldsymbol{R}^d$.

Nearest neighbor search for a dataset D is described as follows. Given a query point q in $\boldsymbol{R}^d$, we want to search a nearest point of q in D where the nearness of two points in D is defined with respect to the Euclidean distance in $\boldsymbol{R}^d$. In

the following, the Euclidean distance between two data points, p and q, in $\boldsymbol{R}^d$ is denoted as $dist(p, q)$.

2.2 Flexible Distance-Based Hashing (FDH)

Flexible distance-based hashing (FDH) is a hashing method proposed in [8] for nearest neighbor search in a high-dimensional space. Given a dataset $D = \{p_1, \ldots, p_n\}$, the method firstly selects a set of A points, denoted $A_n = \{a_1, a_2, \ldots, a_A\}$, from D so that the distance between any pair of a_i and a_j, $i \neq j$, is large as much as possible. Each a_i is called an *anchor*. For each anchor a_i in A_n, we determine the "radius", denoted r_i, so that $|D_{near}(r_i)| \simeq |D_{far}(r_i)|$ holds where $D_{near}(r_i) = \{p \in D | dist(a_i, p) \leq r_i\}$ and $D_{far}(r_i) = \{p \in D | dist(a_i, p) > r_i\}$.

Determining a set of anchors appropriately is important to get a good solution in the nearest neighbor search using FDH. A heuristic approach to this problem is an iterative improvement method. In this method, firstly, a set of anchors is randomly selected. Next, the method selects a pair of distinct anchors, p and q, such that $dist(p, q)$ is the smallest among all pairs of anchors. Then, another point, r, is randomly selected, and if the minimum distance between r and any other point is larger than $dist(p, q)$, p is discarded from the set of anchors, and r becomes an element of the set of anchors. Otherwise, r is discarded. The method repeats this procedure until no more improvement is observed.

Assume that we have an anchor set A_n for a dataset D in $\boldsymbol{R}^d$. Given a data point p in $\boldsymbol{R}^d$, a bit sequence of length A, denoted $BM(p) = b_1, b_2, \ldots, b_A$, is defined as follows.

$$BM(p)[i] = b_i = \begin{cases} 0 \; if \; dist(a_i, p) \leq r_i \\ 1 \; otherwise \end{cases} \tag{1}$$

We call this bit sequence $BM(p)$ the *bitmap* of data point p. Using the bitmap, the d-dimensional data space can be divided into a set of 2^A subspaces. That is, two distinct data points, p and q, are included in the same subspace if $BM(p) = BM(q)$ holds. In the following, we call each subspace a *region*. A region can be identified by its corresponding bitmap. For example, when $d = 2$ and $A = 3$, then the 2-dimensional data space is divided into 8 regions, whose boundaries are represented as 3 circles (see Fig. 1).

For a dataset D in the d-dimensional data space, if each data in D is maintained in its corresponding region, we can efficiently search a nearest point for a given query point q by calculating $BM(q)$. This search method has been firstly proposed in [8], and named *Flexible Distance-based Hashing (FDH)*. For example, in Fig. 1, there are 16 data points, p_1, p_2, ..., p_{16}, in which $p_{14} = a_1$, $p_{15} = a_2$, and $p_{16} = a_3$ are anchors. If a query point p is given, its bitmap is 010 ($BM(p) = 010$). The region with bitmap 010 includes p_5, p_7, p_8, and p_{11}. We calculate the Euclidean distance between p and those 4 data points, and find p_8 as the closest point to p.

Since if two data points exist in a neighborhood in the data space, there is a high possibility that two data points belong to a same region (i.e., they have a

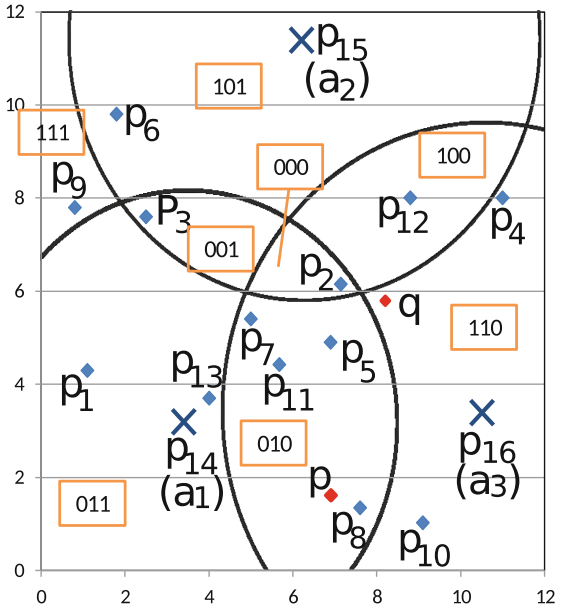


Fig. 1. An example of FDH.

same bitmap), FDH can be considered as a family of locality sensitive hashing. An advantage of FDH is that, since there is no parameters of FDH depending on the number of dimensions of data space, the nearest neighbor search using FDH can be easily applied to any number of dimensions without degrading its performance.

2.3 Nearest Neighbor Search Using FDH

FDH can be used for nearest neighbor search. However, a straightforward, brute-force approach to nearest neighbor search could fail to find an exact solution, since there is no guarantee that a region including a query point includes the nearest point to the given query. To relax this problem, the concept of neighbor regions is introduced. Given two data points, p and q, $Hamming(p, q)$ represents the Hamming distance of bitmaps of p and q. For example, if $BM(p) = 011$ and $BM(q) = 110$, then $Hamming(p, q) = 2$. We extend the notation of $Hamming()$ so that $Hamming(p, R)$ represents the Hamming distance of $BM(p)$ and $BM(R)$ where p is a data point and R is a region, and $BM(R)$ represents the bitmap of R.

The nearest neighbor search using FDH is given as follows. Assume that a positive integer H is given as an input, and $H \leq A$ holds where A is the number of anchors in FDH. Then, a given query p, a set of *neighbor regions*, R_1, R_2, ..., R_k is determined so that for each R_i, $1 \leq i \leq k$, $BM(p, R_i) \leq H$ holds. A nearest neighbor is searched in those neighbor regions, and a data point closest

to the query is output as a search result. There is still no guarantee that the exact solution is obtained. However, the accuracy of search can be controlled by the value of H. Note that, when $H = A$, then this search algorithm is equivalent to the full search algorithm, and the obtained result becomes an exact solution.

3 The Proposed Algorithm

3.1 A Drawback of Nearest Neighbor Search Using FDH

In Subsect. 2.3, we described the approximate nearest neighbor search algorithm using FDH. As explained, this algorithm has a possibility that an exact solution is failed to find. For example, consider Fig. 1 again. In this figure, there are 16 data points including 3 anchors. Assume that a query point q is given. A region having q has bitmap 110. In the following, a region having the query point is called a *query region*. In the query region, there are 3 data points, but any of those are very far from an exact solution p_2. If the Hamming distance H is given as 1, which specifies neighbor regions to the query, bitmaps of neighbor regions are 010, 100, 111, but those regions still don't contain p_2. If H is given as 2, the region containing p_2 becomes a neighbor region, and an exact solution can be found. However, in this case, the number of neighbor regions including the query region is 7, and all regions but one (region 001) are to be searched for finding a nearest neighbor. This is a drawback of nearest neighbor search using FDH. To solve this drawback, two mechanisms are newly introduced, that is, a furthest δ-neighbor and an adaptive control of neighbor regions, which will be explained in the subsequent subsections.

3.2 Furthest δ-Neighbor

Let's pay attention to Fig. 1 again. The reason that an exact result p_2 for query point q is failed to find when $H = 1$ is that the query point q exists very near to spheres (circles) defined by two anchors a_1 and a_2. In this case, region with bitmap 000 is far from the query region 110 in the Hamming distance (the Hamming distance from the query to the region is 2), but in the Euclidean distance, region with bitmap 000 is near to the query. An idea to solve this drawback is to add such regions like region 000 in the figure to neighbor regions to be searched in the search algorithm using FDH.

Given a query q and a positive real number less than 1.0, denoted δ, we newly introduce a furthest δ-neighbor, denoted $FN(q, \delta)$, to the search algorithm. A furthest δ-neighbor is a region which is near to the query in the Hamming distance, but furthest in the Euclidean distance. A furthest δ-neighbor $FN(q, \delta)$ is defined as follows. Let the bitmap of q be $BM(q) = b_1, b_2, \ldots, b_A$. For each anchor a_i, $1 \leq i \leq A$, a bitmap $b'_1, b'_2, \ldots, b'_A$ is defined as

$$
\begin{aligned}
&if\ (dist(a_i, q) \leq r_i) \wedge (dist(a_i, q) \geq (1 - \delta) r_i)\ then\ b'_i = 1 \\
&if\ (dist(a_i, q) > r_i) \wedge (dist(a_i, q) \leq (1 + \delta) r_i)\ then\ b'_i = 0 \\
&otherwise\ b'_i = b_i
\end{aligned}
$$

where r_i is a radius corresponding to anchor a_i, and $dist(a_i, q)$ represents the Euclidean distance between a_i and q. Then a furthest δ-neighbor of q, $FN(q, \delta)$, is a region whose bitmap is $b'_1, b'_2, \ldots, b'_A$ defined as above (that is, $BM(FN(q, \delta)) = b'_1, b'_2, \ldots, b'_A$). For example, in Fig. 1, let $\delta = 0.1$, and a query q is given. Then, $BM(q) = 110$. According to the definition of bitmap of furthest δ-neighbor, $b'_1 b'_2 b'_3 = 000$. Thus, $FN(q, \delta)$ is a region with bitmap 000.

Introducing the furthest δ-neighbor, the nearest neighbor search algorithm using FDH is modified as follows. Given a query q, H and δ, we firstly calculate a set of neighbor regions, denoted $NR(q, H)$, which are defined as same as the original search algorithm using FDH. That is, $NR(q, H)$ is formally given as follows.

$$NR(q, H) = \{R \mid R \in R_{all}, Hamming(q, R) \leq H\}$$

where R_{all} is a set of all regions in the data space. For example, in Fig. 1, for q and $H = 1$, $NR(q, H) = \{110, 010, 100, 111\}$, where regions are represented by their corresponding bitmaps (this representation will be used hereafter). Note that, as explained, if $H = A$, $NR(q, H)$ represents all regions of the data space, and the search algorithm becomes a brute-forth, full search algorithm.

Then, we calculate additional neighbor regions, denoted $FNR(q, \delta)$, based on the the furthest δ-neighbor, given as follows.

$$FNR(q, \delta) = \{R \mid R \in R_{all}, Hamming(q, R) + Hamming(R, F) \leq Hamming(q, F)\}$$

where F is the furthest δ-neighbor region of q, that is, $F = FN(q, \delta)$. The intuitive explanation for the inequality in the definition is that R exists somewhere between q and F, but not beyond F. Figure 2 shows a schematic depiction of $NR(q, H)$ and $FNR(q, \delta)$.

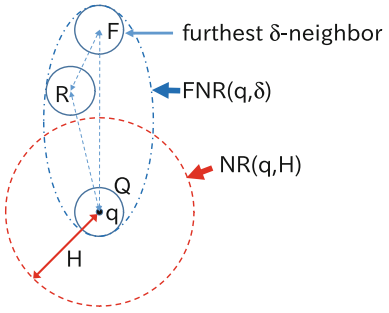


Fig. 2. Furthest δ-neighbor.

For example, in Fig. 1, for q and $\delta = 0.1$, $F = 000$, and $FNR(q, \delta) = \{000, 010, 100, 110\}$. Other regions are not included in this set, since the inequality in the definition does not hold. We can easily show that the number of regions in $FNR(q, \delta)$ is shown as follows.

$$|FNR(q,\delta)| = 2^{Hamming(q,F)}$$

where F is the furthest δ-neighbor region of q.

Then, a candidate of a nearest neighbor will be searched in $NR(q,H) \cup FNR(q,\delta)$. Note that there are some regions which belong to both $NR(q,H)$ and $FNR(q,\delta)$. In the case of Fig. 1, regions of $\{000, 010, 100, 110, 111\}$ are searched. In this example, exact solution p_2 will be found. Note that the same result could be obtained with the original search algorithm with $H = 2$. But, in this case, 7 regions will be searched, and hence the computation time would be larger than the improved algorithm given in this section.

3.3 Adaptive Control of δ

The search algorithm presented in the previous subsection is superior to the original search algorithm using FDH. However, there is still one issue of the improved algorithm. That is, the effectiveness of the algorithm would depend on the values of parameters. The algorithm has two parameters, H and δ. Preliminary experiments show that the accuracy of results as well as computation time vary as the parameter values are changed. Experimental results show that, in general, H and δ are set to large values, the accuracy of results becomes high, but requires more computation time. For a given data set and a query, it is difficult to set appropriate values to the parameters in advance.

To overcome this difficulty, we introduce the mechanism of adaptive control of the value of δ during the search. The overview of the adaptive control of parameter value is as follows. When the algorithm starts, δ is set to 0. This means that the behavior of the algorithm is the same as the original search algorithm using FDH. After the algorithm finishes, the obtained result is temporarily stored. Then, the value of δ set to a predefined value, denoted Δ, and the algorithm starts again. After finishing the algorithm, if the obtained result is improved, we update δ as $\delta \leftarrow \delta + \Delta$, and the algorithm runs again. Otherwise, the algorithm terminates. We call the modified FDH explained in this section the adaptive FDH. As shown later in Sect. 4, experimental results show that the proposed mechanism works very well to get a good solution.

3.4 Algorithm Description

The proposed approximate nearest neighbor search algorithm, which includes furthest δ-neighbor and adaptive control of the value of δ, is given below.

Inputs: a query: q, a data set: $D = \{p_1, p_2, \ldots, p_n\}$, a set of anchors: $A_n = \{a_1, a_2, \ldots, a_A\}$, a set of radius: $R_a = \{r_1, r_2, \ldots, r_A\}$, the maximum Hamming distance: H, the incremental value of δ: Δ.
Step1: Calculate $BM(q)$.
Step2: $\delta \leftarrow 0$, $SR \leftarrow NR(q,H)$, $min_d \leftarrow \infty$.
Step3: $old_d \leftarrow min_d$.
Step4: For each $p \in SR$ do:

Step4-1: If $dist(q, p) < min_d$ then $p_{min} \leftarrow p$, $min_d \leftarrow dist(q, p)$.
Step4-2: If there is no data point left in SR, then go to Step5, otherwise go to Step4.
Step5: If $old_d = min_d$ then output p_{min}, stop.
Step6: $\delta \leftarrow \delta + \Delta$.
Step7: $SR \leftarrow$ {regions in $FNR(q, \delta)$ newly added by increment of δ}, go to Step3.

The main advantage of the proposed nearest neighbor search algorithm using adaptive FDH is that there is no need to determine the values of parameters in advance. Although H and Δ are listed in inputs of the algorithm shown above, as shown in experimental results given in Sect. 4, in practice, their values can be fixed in advance, and satisfactory results can be obtained.

4 Experimental Evaluation

We show some experiments to evaluate the proposed search algorithm from the viewpoint of search accuracy. Given a query q, let p_{algo} be a data point obtained by the search algorithm as an approximate nearest neighbor, and p_{opt} be a exact nearest neighbor of q. We define two measures, the *absolute error* $AE(q, p_{algo}, p_{opt})$ and the *relative error* $RE(q, p_{algo}, p_{opt})$, to evaluate the quality of search. The absolute error is defined as follows:

$$AE(q, p_{algo}, p_{opt}) = dist(p_{algo}, q) - dist(p_{opt}, q) \tag{2}$$

The relative error is 0 if the absolute error is 0. For the case that the absolute error is not 0, the relative error is defined as follows:

$$RE(q, p_{algo}, p_{opt}) = (AE(q, p_{algo}, p_{opt})/dist(p_{opt}, q)) \times 100\ (\%) \tag{3}$$

We use another measure for evaluation of search. Consider that the nearest neighbor search is performed in N times, and we get exact nearest neighbors in M times ($M \leq N$). Then, the *accuracy rate*, denoted AR, is defined as $AR = (M/N) \times 100\ (\%)$. To obtain an exact solution, we develop an exact nearest neighbor search program.

As inputs of the search program, we randomly generate 2 sets of datasets. Each dataset consists of $16,384$ data points, and each data point in the dataset has 16 and 32 attributes. Attribute values are real numbers within the range of $(-999.99, 999.99)$.

In experiments, the number of anchors A in AFDH is set to 10, and the upper bound of Hamming distance used in the search to determine regions to be searched, H, is set to 1, 2, 3 and 4. The total number of nearest neighbor search for each data set, N, is $100,000$. Queries are randomly generated.

Experimental results were shown in Tables 1 and 2, each of which show the results of the cases of $d = 16$ and $d = 32$, respectively, where d is the number of attributes of a data point (that is, the number of dimensions). In the tables, "Abs.

Table 1. Experimental results of AFDH (case of $d = 16$).

H	Abs. Err. (Ave.)	Abs. Err. (Max.)	Rel. Err. (Ave.) [%]	Rel. Err. (Max.) [%]	Accuracy [%]	#regions (Ave.)	Time (Ave.) [μs]
$\Delta = 0.1$							
1	101.39	1108.41	7.70	146.32	43.51	51	85.52
2	42.30	951.97	3.20	133.88	67.68	74	126.23
3	14.99	761.33	1.12	77.48	85.98	183	237.26
4	4.22	605.21	0.31	63.69	95.48	388	413.46
$\Delta = 0.2$							
1	46.26	1040.39	3.51	146.32	70.59	300	295.58
2	19.51	951.97	1.47	133.88	84.03	213	318.33
3	6.61	761.33	0.49	77.48	93.57	240	414.86
4	1.74	510.47	0.13	44.31	98.10	404	524.37
$\Delta = 0.3$							
1	11.21	1001.23	0.84	105.46	92.63	588	514.59
2	4.78	878.47	0.36	86.02	96.03	373	507.25
3	1.62	724.32	0.12	66.45	98.40	301	604.73
4	0.42	402.88	0.03	33.43	99.52	419	679.92

Err." and"Rel. Err." represent the absolute and the relative errors, respectively. "Ave." and"Max." represent the average and the maximum values, respectively. "#regions (Ave.)" shows the average number of searched regions during the algorithm execution."Time (Ave.)" represents the average computation time to get a solution per one query. Note that since $A = 10$, the number of all regions is $2^{10} = 1024$.

From experimental results, we have several observations shown as follows. Firstly, as the value of H increases, better results will be obtained. Secondly, as the value of Δ increases, better results will also be obtained. Thirdly, the number of attributes of a data point increases, results will be improved. Fourthly, there always exist very difficult cases, in which exact results can be hardly obtained, but those cases are very rare, in particular, when H and Δ are large. Note that, except for the case that the behavior of the proposed algorithm is the same as a brute-forth full search algorithm (for example, a case of setting H to A), we can show that there always exists a query, for which the proposed algorithm fails to produce an exact solution.

Finally, the best values of parameters of H and Δ depend on the required accuracy. For example, let's consider the case that the accuracy of 99% or more is required. Then, for the case of $d = 16$, $H = 4$ and $\Delta = 0.3$ are the best values in terms of the accuracy and the computation time per query. For the case of $d = 32$, $H = 1$ and $\Delta = 0.3$ are the best values.

Table 2. Experimental results of AFDH (case of $d = 32$).

H	Abs. Err. (Ave.)	Abs. Err. (Max.)	Rel. Err. (Ave.) [%]	Rel. Err. (Max.) [%]	Accuracy [%]	#regions (Ave.)	Time (Ave.) [μs]
$\Delta = 0.1$							
1	102.85	1366.22	3.87	81.47	47.32	175	262.24
2	61.83	1205.84	2.32	61.73	61.15	160	323.02
3	30.16	1055.92	1.13	54.98	77.51	228	431.75
4	11.92	1055.92	0.44	54.98	89.92	406	662.41
$\Delta = 0.2$							
1	17.27	1157.59	0.65	57.44	89.61	696	679.20
2	10.45	1055.92	0.39	54.98	92.69	539	785.94
3	5.08	1055.92	0.19	54.98	95.97	429	877.87
4	1.98	1055.92	0.07	54.98	98.25	478	1043.75
$\Delta = 0.3$							
1	0.86	746.28	0.03	33.69	99.45	836	834.92
2	0.51	597.85	0.02	26.49	99.62	642	843.50
3	0.26	527.91	0.01	26.49	99.79	483	976.72
4	0.09	399.78	0.00	13.95	99.91	497	1103.41

In addition, for both cases, $H = 4$ and $\Delta = 0.3$ always achieve the accuracy of 99% or more. This result shows that, in practice, there is no need to adjust the parameter values of the proposed algorithm, although fine tuning of parameter values will result better performance as well as better results of the algorithm. Depending on the requirement of the application, we can choose appropriate parameter values so that better results can be obtained in shorter computation time.

5 Conclusion

In this paper, we proposed a hashing-based approximate nearest neighbor algorithm for high-dimensional data. The search algorithm was based on flexible distance-based hashing (FDH). We extend FDH by introducing the concept of δ-nearest neighbor and the adaptive control of parameter value δ used in the algorithm, and call this extended FDH the adaptive FDH. Using AFDH, we propose a nearest neighbor search algorithm for high-dimensional data. Experiments show that the proposed search algorithm was effective to obtain a good result. The proposed algorithm can produce good results even if there is no parameter tuning of the algorithm.

As future work, further improvement of the search method using FDH may be desired so that better approximate results can be obtained in a shorter computation time. Comparing the proposed algorithm with existing nearest neighbor

algorithms is interesting and important. Experiments using benchmark data as well as real data will be required to show the effectiveness of the proposed algorithm in the real world applications. Extension of the algorithm to a k-nearest neighbor algorithm is also attractive.

References

1. Athitsos, V., Potamias, M., Papapetrou, P., Kollios, G.: Nearest neighbor retrieval using distance-based hashing. In: Proceedings of IEEE International Conference on Data Engineering, pp. 327–336 (2008)
2. Biau, G., Devroye, L.: Lectures on the Nearest Neighbor Method. SSDS. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25388-6
3. He, J., Radhakrishnan, R., Chang, S.-F., Bauer, C.: Compact hashing with joint optimization of search accuracy and time. In: Proceedings of 2011 IEEE Conference on Computer Vision and Pattern Recognition, pp. 753–760 (2011)
4. Hinneburg, A., Aggarwal, C.C., Keim, D.A.: What is the nearest neighbor in high dimensional spaces? In: Proceedings of 26th VLDB Conference, pp. 506–515 (2000)
5. Hwang, Y., Han, B., Ahn, H.-K.: A fast nearest neighbor search algorithm by nonlinear embedding. In: Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3053–3060 (2012)
6. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 2227–2240 (2014)
7. Yianilos, P.N.: Data structures and algorithms for nearest neighbor search in general metric spaces. In: Proceedings of Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 311–321 (1993)
8. Yiu, M.L., Assent, I., Jensen, C.S., Kalnis, P.: Outsourced similarity search on metric data assets. IEEE Trans. Knowl. Data Eng. **24**(2), 338–352 (2012)

Approximate Set Similarity Join Using Many-Core Processors

Kenta Sugano[1(✉)], Toshiyuki Amagasa[2], and Hiroyuki Kitagawa[2]

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Japan
sugano@kde.cs.tsukuba.ac.jp

[2] Center for Computational Sciences, University of Tsukuba, Tsukuba, Japan
{amagasa,kitagawa}@cs.tsukuba.ac.jp

Abstract. Set similarity join is a database operation used to find out all similar pairs of sets from two collections over sets. Due to the high versatility, it has been applied in many applications in various domains, including text processing, image processing, etc. One of its drawbacks is the high computational costs, especially when the size of the given collections is large. In this paper, we propose a scheme for efficient set similarity join on Intel Xeon Phi, one of the latest many core processor. In order to make best use of high computational power of Intel Xeon Phi, we employ following approaches: (1) we compress each record by b-bit MinHash to fit them in MCDRAM which is a high bandwidth on-chip memory; and (2) we apply some optimizations such as utilization of 512-bit SIMD instructions and loop unrolling. Experimental results show that our proposed method outperforms CPU implementation.

Keywords: Intel Xeon Phi · Set similarity join · MinHash

1 Introduction

Set similarity join is a database operation used to find out all similar pairs of sets from two collections of sets. It has many practical applications such as data cleaning, entity resolution, image matching, etc., where an object can be represented as a set of features. One of its problem is that it is demanding for both time and space, which makes it difficult to apply it to large datasets. This is due to the fact that it basically requires pair-wise comparisons over all possible combinations of objects (or records) consisting of many features.

To cope with this problem, there have been several research works that attempt to approximately compute similarity between sets. Among others, MinHash and its variants [1,6,7] are the most widely recognized algorithms. Instead of make exhaustive comparisons over set items, for each set, it extracts some features whereby similarity computation can be done by comparing the features. Notice that, once the features are extracted, the original set is no longer necessary, thereby allowing us to save the memory space and computation time as well. One may afraid about the fact that the final result is approximated; i.e.,

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 214–222, 2018.
https://doi.org/10.1007/978-3-319-98812-2_18

the results may contain false positives and/or false negatives. However, in real applications, it is often the case that we do not necessarily stick to the correctness of the results. Nevertheless, it is still important to speed up its computation in particular when dealing with very large datasets.

In the meantime, many-core processors have been gaining much attentions due to their high computational power. It has become common that high-end processors more than 16 processing cores. In addition, Intel Xeon Phi, a series of many-core processors, has emerged and applied in different fields such as HPC (High Performance Computing) and big data analytics. Xeon Phi has many physical cores and VPUs (Vector Processing Units) which support 512 bit SIMD instructions, thereby achieving high computational power for data intensive workloads [4,9].

So far, there have been many researches to accelerate set-similarity join from different approaches. Xiao et al. proposed some filtering methods to eliminate unnecessary comparisons thereby achieving better performance while maintaining accurate result [12]. From another direction, parallel processing has been successfully applied by using shared-nothing clusters [10,11], multi-core processors [5], and GPU [3,8]. However, none of the existing works has applied the latest many-core processor, i.e., Xeon Phi, for accelerating approximate similarity join processing.

In this paper, we propose a new scheme for efficient approximate set similarity join on Intel Xeon Phi. We chose Intel Xeon Phi because GPU-based approach has been studied by other works [3]. Besides, Intel Xeon Phi as the following advantages against GPU: (1) the processing cores share the same memory space whose space is much greater than that of GPU, while we need to transfer necessary data between main memory and GPU's device memory, which incurs extra cost; and (2) we can enjoy ordinary programming language, e.g., C and C++, while we need to learn a dedicated programming environment, e.g., CUDA or OpenCL, to use GPU. On the other hand, in order to make the best use of high computational power of Intel Xeon Phi, two challenges need to be addressed: (1) hiding memory access latency; and (2) efficient parallelization utilizing many physical cores and 512-bit SIMD instructions. To address these challenges, we employ b-bit MinHash [6], which is a space efficient algorithm to estimate the Jaccard similarity between two sets. In this algorithm, each set is converted to a compact data structure called signature. Therefore, we can fit large-scale collections in MCDRAM and cache memory as well, thereby enabling us to hide memory access latency. In addition, we apply two optimizations: SIMD Vectorization and loop unrolling. Experimental results show that our proposed method is faster than CPU implementation, while maintaining accuracies applicable in many applications.

2 Preliminaries

2.1 Intel Xeon Phi

Intel Xeon Phi is a series of many-core processors. The main hardware features of the second generation Intel Xeon Phi, called KNL (Knights Landing), are the

followings: (1) it has up to 72 cores and four hardware threads per core; (2) it supports 512-bit SIMD instructions; and (3) it has high bandwidth memory called MCDRAM (Multi-Channel DRAM) in addition to the DDR4 memory.

The key points to achieve high performance on KNL are utilization of these features by efficient use of multithreading and SIMD instructions. In addition, since the memory access latency is significantly larger than that of ordinary CPUs, it is also important to exploit CPU cache as much as possible and MCDRAM to hide the latency.

2.2 Set Similarity Join

Set similarity join is defined as follows:

Definition 1. *Given two collections of sets R and S, a set similarity function* $\mathrm{Sim}()$*, and a similarity threshold t, set similarity join is defined as* $R \bowtie_t S \triangleq \{\langle r, s\rangle \in R \times S \mid \mathrm{Sim}(r, s) \geq t\}$.

We suppose that the Jaccard similarity $\mathrm{Jac}(r, s) = |r \cap s|/|r \cup s|$ is used as the set similarity function and threshold $t = 0.65$. Set similarity join returns the pair $\langle r_1, s_3\rangle$ as $\mathrm{Jac}(r_1, s_3) = 2/3 \geq t$ and all the others are not similar. Henceforth, we use collection to denote collection of sets, if there is no ambiguity.

There are some commonly used similarity functions between sets such as Jaccard, Dice, and cosine similarities. In this paper, we use Jaccard similarity as the set similarity function. There are two drawbacks of calculating Jaccard similarity exactly. First, It requires high computational cost due to a number of pairwise comparisons of two elements from each set. Second, it is necessary to store the whole sets in memory.

2.3 MinHash

To address the above problems, Broder et al. proposed MinHash [1]. It compresses each set to a *signature* and estimates the Jaccard similarity using the *signatures* without accessing the original sets. A *signature* is defined as followed:

Definition 2. *Given a set $r \subseteq \Omega = \{0, .., D-1\}$ and hash functions $h_1, .., h_k : \Omega \mapsto \Omega$, the signature of r is defined as* $\mathrm{Sig}(r)$=$\langle \min_{e\in r}(h_1(e)), .., \min_{e\in r}(h_k(e))\rangle$.

We can estimate the Jaccard similarity between r and s by $\hat{\mathrm{Jac}}(r, s) = 1 - \mathrm{Ham}(\mathrm{Sig}(r), \mathrm{Sig}(s))/k$, where Ham() represents Hamming distance between two *signatures*. There is a tradeoff relation between estimation accuracy and computational cost; the larger the parameter k, the higher the estimation accuracy. Conversely, the larger the size of the *signatures* and the longer the execution time.

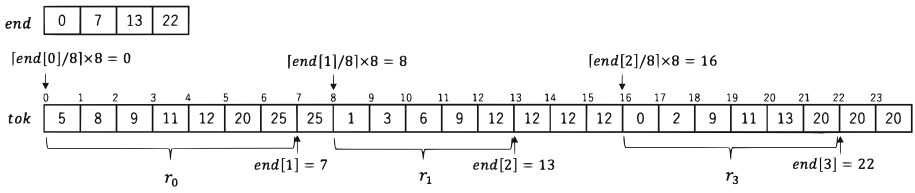


Fig. 1. An example of array representation.

2.4 b-bit MinHash

Li et al. proposed b-bit MinHash [6] to improve the trade-off of original MinHash. Instead of storing entire minimum hash value for each hash function, it stores only the lowest b bits. Although it causes increasing unexpected hash collisions, it is possible to maintain the estimation accuracy by increasing the number of hash functions k, thereby achieving high space efficiency compared with the original MinHash. The Jaccard similarity between r and s can be estimated by

$$\hat{\mathrm{Jac}}(r,s) = 1 - \frac{\mathrm{Ham}\left(\mathrm{Sig}_b\left(r\right), \mathrm{Sig}_b\left(s\right)\right)}{\left(1 - 2^{-b}\right)k} \tag{1}$$

where $\mathrm{Sig}_b()$ represents a *signature* of b-bit MinHash.

One of its important characteristics is that we can calculate it efficiently because calculation of Hamming distance between two *signatures* can be implemented efficiently with SIMD instructions, in particular when $b = 1$. We will elaborate the detail in the next section.

3 Proposed Method

In this section, we describe our proposed method for efficient approximate set similarity join for Intel Xeon Phi. As we mentioned in Sect. 2.1, it is important to be aware of the cache architecture and MCDRAM, so we employ b-bit MinHash to fit large-scale collections in them. Hereafter, we denote by set similarity join the approximated version of set similarity join by b-bit MinHash if there is no ambiguity.

Our proposed method comprises two phases as similar to other MinHash-based algorithms: (1) compression phase and (2) join phase. In the compression phase, we compress each set (or record) in the given collections to *signature* by b-bit MinHash. Then, in the join phase, similarity join is performed by estimation of Jaccard similarity using the b-bit MinHash *signatures*. Notice that we regard compression phase as a offline process, which means that collections are compressed and stored in disk storage before join queries arrive.

3.1 Array Representation of Collections

We assume that a collection is represented by the following two arrays:

– *tok*: This array stores the elements of each set in the collection. Each element is 4-byte integer and the first element of each record is aligned on a 64-byte boundary. The space between the last element of the i-th set and the first element of the (i+1)-th set is padded with the last element of the i-th set.
– *end*: This array stores the position in *tok* of the last element of each set.

Algorithm 1. Parallel b-bit MinHash

input : collections of sets $R = \{r_1, ..., r_n\}$, hash functions $h_1, ..., h_k : \Omega \mapsto \Omega$
output: collections of signatures $R' = \{r'_1, ..., r'_n\}$

```
for i = 1 to n in parallel do // Executed by threads
  for j = 1 to k do
    val ← min_{e∈r_i} (h_j (e))
    Store the lowest bit of val to j-th bit of r'_i
  end for
end for
```

Algorithm 2. Parallel join

input : collections of signatures $R' = \{r'_1, ..., r'_n\}, S' = \{s'_1, ..., s'_m\}$, threshold t_h
output: Output space O

```
for i = 1 to n in parallel do // Executed by threads
  for j = 1 to m do
    if popcnt(r'_i ^ s'_j) ≤ t_h then
      Add < i, j > to O
    end if
  end for
end for
```

Since it is possible to quickly access data aligned on a 64-byte boundary in KNL architecture, the array representation enables fast access to the elements of each set. Figure 1 shows an example of array representation where each set in *tok* is aligned on a 32-byte boundary. Note also that this representation is applicable to various types of data including text, transaction, image features, etc., although we will test the performance using text dataset.

3.2 Compression Phase

In this phase, we compress each set in a collection represented by two arrays to a *signature* using b-bit MinHash. As we mentioned in Sect. 2.4, we can calculate the estimate value quickly when $b = 1$, so we set $b = 1$, which means that we store only the lowest bit of the minimum hash value for each hash function. In addition, we set the power of 2 between the range of 32 to 512 as k so that we utilize 512-bit SIMD instruction effectively in the join phase.

Algorithm 1 shows how we compress a collection in parallel. We divide the collection to sets each of which corresponds to a string and assign a thread to each set (Line 1). Each thread is responsible for compressing the assigned set into *signature* (Lines 2–5). Since the process for each set is independent of each other, we can parallelize it efficiently. Finally, *signatures* are stored in the disk storage.

3.3 Join Phase

Algorithm 2 shows how we perform join using b-bit MinHash *signatures*. First, the *signatures* are read from the disk storage and are stored in the space aligned on the 64-byte boundary in MCDRAM. Then, we decompose the *signatures* R' and assign each to a thread (Line 1). Each thread is responsible for joining the assigned signatures of R' and the whole of S' (Lines 2–6). The operator ˆ in Line 3 represents bit-wise XOR and popcnt() in the same line represents the function to get the number of non-zero bits in the given integer. Thus, we calculate the Hamming distance between two *signatures* in Line 3. After that, we output the pair of sets if the Hamming distance between corresponding two *signatures* is smaller than t_h, which is the threshold of Hamming distance between two *signatures* which is obtained by transforming the Eq. 1.

3.4 Further Optimizations

SIMD Vectorization. To take advantage of 512-bit SIMD instructions, we vectorize the processing by using Intel AVX512 which is a SIMD instruction set available on KNL.

In the compression phase, we vectorize Line 3 in Algorithm 1. We group the elements of a set into 16 pieces and calculate hash values simultaneously for each group. Since each set is aligned on 64-byte boundary in *tok*, we can load them into SIMD registers quickly. In addition, we get the minimum value by reduction of the hash values by using SIMD instructions.

In the join phase, we vectorize Lines 2–6 in Algorithm 2. We group *signatures* of S' into several pieces. Then, we calculate Hamming distance and compare it with the threshold simultaneously for each group by using SIMD instructions. The number of *signatures* in each group depends on the size of *signatures*. In the case of the size of *signature* is 32-bit, *signatures* are grouped into 512/32=16 pieces. Since each group of *signatures* is aligned on 64-byte boundary, we can load them into SIMD registers quickly.

Loop Unrolling. Loop unrolling is a technique to reduce the loop overhead by reducing the number of iterations and replicating the body of the loop. Furthermore, the degree of instruction level parallelism can be improved by appropriately combining the replicated instructions. We unroll the loop of Line 1 of Algorithm 2, which reduce the number of load instruction as well.

4 Experiments

In this section, we present the experimental evaluations. The objective is to check the effect of the optimizations and compare the performance with normal CPU implementation.

4.1 Environment

In our experiments, we chose datasets from two distinct domains, i.e., real and artificial datasets. *Enron*[1] is composed of 517,431 emails and we tokenize each body into words. *Large* is composed of 5,000,000 artificially generated texts. The number of words in each text follows the normal distribution N(140,40), and we randomly selected words in each text from 235,886 words in Unix's dictionary file (/usr/share/dict/words).

The CPU used in our experiments is an Intel Xeon E5-1620 v3 (4 cores, 8 threads). The Xeon Phi is an Xeon Phi 7250 (68 cores, 272 threads) with 16 GB MCDRAM. We used MCDRAM in flat mode and mapped all memory requests to MCDRAM using the numactl utility. We used Intel C++ Compiler with -O3 option and we implemented multi-threading by OpenMP 4.0.

We experimentally obtained that both the precision and recall reached about 95% when $k = 32$. Hence, we set $k = 32$ unless explicitly specified henceforth.

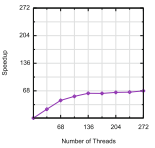


Fig. 2. Scalability

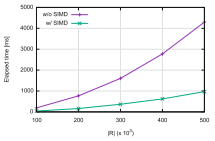


Fig. 3. Acceleration by SIMD instructions.

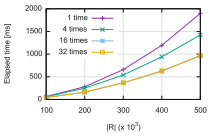


Fig. 4. Acceleration by loop unrolling.

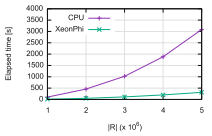


Fig. 5. Comparison with CPU

4.2 Performance Evaluation

Scalability. We evaluated the speedups of the join phase using *Enron* as shown in Fig. 2. The speedup is the ratio of the elapsed time of the parallel program with that of the single thread program. We can see that our method achieved 67.9x faster processing when the number of threads is 272, where all physical cores runs 4 threads simultaneously. This result shows that our method achieves good scalability.

[1] http://www.cs.cmu.edu/~enron/.

SIMD Vectorization. We evaluated the acceleration of the process by SIMD instructions using *Enron*. Figure 3 presents the elapsed time of the join phase with and without SIMD instructions. We can see that the join with SIMD instructions achieved up to 4.6x faster than the one without instructions. In the compression phase, SIMD instructions achieved up to 7.9x faster. Since branch mispredictions occur in the join phase, the speedup rate of the join phase is relatively low.

Loop Unrolling. We evaluated the acceleration of the join phase by loop unrolling using *Enron*. Figure 4 presents the elapsed time of the join phase applied the loop unrolling. We can see that we achieved speedups of approximately 2.0x when the loop unrolled 16 times due to reducing the loop overhead and the number of load instructions.

Comparison with CPU. We compared our method with the CPU implementation which is parallelized by 8 threads with SIMD instructions. We used *Large* as dataset. Figure 5 shows the comparison in the join phase. We can see that our method achieved up to 9.8x faster than the CPU implementation. This result shows that significant speedup was achieved by using Xeon Phi.

5 Related Work

Existing works are roughly classified into two groups. The first one reduces the number of candidate pairs, which we need to carry out similarity calculation, adopting filtering techniques [2,12]. PPJoin [12] adopts several powerful filtering techniques such as *Prefix filtering* and *Positional filtering*, and reduces significant amounts of computational cost.

The second one exploits paralell computation using PC clusters [10,11], GPUs [3], or multi-core processors [5]. Cruz et al. proposed a scheme of set similarity join for GPU [3]. In order to perform the join on a limited memory space of GPU, it uses One Permutation Hashing [7] which is one of the improvements of MinHash.

6 Conclusions

We proposed a set similarity join scheme that uses b-bit MinHash to exploits Intel Xeon Phi and achieved a speedup of up to 9.8x when compared to CPU implementation. The Xeon Phi optimizations such as SIMD vectorization and loop unrolling accelerated the processing significantly.

Our future works will focus on the experiments over large data sets which can not fit in cache memory and MCDRAM, and on adopting filtering techniques exploiting Xeon Phi architecture.

References

1. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. J. Comput. Syst. Sci. **60**(3), 630–659 (2000)
2. Chaudhuri, S., Ganti, V., Kaushik, R.: A primitive operator for similarity joins in data cleaning. In: ICDE (2006)
3. Cruz, M.S.H., Kozawa, Y., Amagasa, T., Kitagawa, H.: Accelerating set similarity joins using GPUs. Trans. Large-Scale Data- Knowl.-Cent. Syst. **28**, 1–22 (2016)
4. Jha, S., He, B., Lu, M., Cheng, X., Huynh, H.P.: Improving main memory hash joins on intel xeon phi processors: an experimental approach. Proc. VLDB **8**(6), 642–653 (2015)
5. Jiang, Y., Deng, D., Wang, J., Li, G., Feng, J.: Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints. In: EDBT/ICDT Workshop (2013)
6. Li, P., König, A.C.: b-Bit minwise hashing. In: WWW (2010)
7. Li, P., Owen, A.B., Zhang, C.-H.: One permutation hashing. In: NIPS (2012)
8. Lieberman, M.D., Sankaranarayanan, J., Samet, H.: A fast similarity join algorithm using graphics processing units. In: ICDE, pp. 1111–1120 (2008)
9. Lu, M., Liang, Y., Huynh, H.P., Ong, Z., He, B., Goh, R.S.M.: MrPhi: an optimized mapreduce framework on intel xeon phi coprocessors. IEEE TPDS **26**(11), 3066–3078 (2015)
10. Metwally, A., Faloutsos, C.: V-SMART-Join: a scalable mapreduce framework for all-pair similarity joins of multisets and vectors. PVLDB **5**(8), 704–715 (2012)
11. Rong, C., Lin, C., Silva, Y.N., Wang, J., Lu, W., Du, X.: Fast and scalable distributed set similarity joins for big data analytics. In: ICDE (2017)
12. Xiao, C., Wang, W., Lin, X., Yu, J.X., Wang, G.: Efficient similarity joins for near-duplicate detection. ACM TODS **36**(3), 151–1541 (2011)

Mining Graph Pattern Association Rules

Xin Wang(✉) and Yang Xu

Southwest Jiaotong University, Chengdu, China
xinwang@swjtu.cn, xuyang@my.swjtu.edu.cn

Abstract. We propose a general class of graph-pattern association rules (GPARs) for social network analysis, *e.g.,* discovering underlying relationships among entities in social networks. Despite the benefits, GPARs bring us challenges: conventional support and confidence metrics no longer work for GPARs, and discovering GPARs is intractable. Nonetheless, we show that it is still feasible to discover GPARs. We first propose a metric that preserves *anti-monotonic* property as support metric for GPARs. We then formalize the GPARs mining problem, and decompose it into two subproblems: frequent pattern mining and GPARs generation. To tackle the issues, we first develop a parallel algorithm to construct *DFS code graphs*, whose nodes correspond to frequent patterns. We next provide an efficient algorithm to generate GPARs by using *DFS code graphs*. Using real-life and synthetic graphs, we experimentally verify the performance of the algorithms.

1 Introduction

Association rules have been studied for discovering regularities between items in relational data [4]. They have a traditional form $X \Rightarrow Y$, where X and Y are disjoint itemsets. There have been recent interests in studying associations between entities in social graphs, *e.g.,* a special kind of graph pattern association rules are introduced in [7]. While, as these rules have consequents taking only a single edge, they are not capable enough to model even more complicated associations among entities in social networks. Nonetheless, GPARs are more involved with generalized patterns as antecedents and consequents. This highlights the need for studying how to discover generalized GPARs on social graphs.

Example 1. A fraction of a social network G is shown in Fig. 1(a), where each node denotes a person with name and job title (e.g., project manager (PM), database administrator (DBA), programmer (PRG), business analyst (BA) and software tester (ST)); and each edge indicates friendship, *e.g.,* (Bob, Mat) indicates that Bob and Mat are friends. One can easily infer the rule from graph G that among a group of people with titles PM, BA, DBA, PRG and ST, if PM and BA, PM and DBA, DBA and PRG, DBA and ST, PRG and ST are friends, then the chances are that PRG and BA, BA and DBA are likely to be friends. As shown in Fig. 1(b), the antecedent and consequent of the rule, which are represented as graph patterns, *i.e.,* Q_l and Q_r, specify conditions on various entities in a social graph in terms

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 223–235, 2018.
https://doi.org/10.1007/978-3-319-98812-2_19

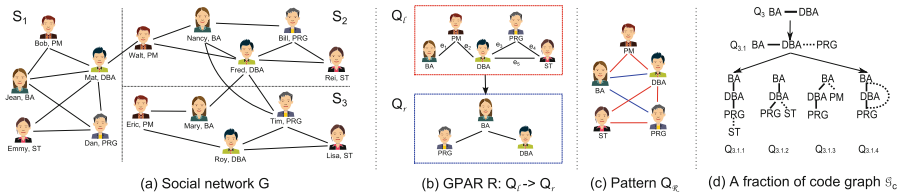


Fig. 1. Graph, association as graph patterns and code graph

of topological constraints, *e.g.,* friendship. With the rule, one can infer social relationships, and recommend friends to others who are most likely interested in, *e.g.,* recommend `Mary` to `Dan`, and `Roy` to `Mary`. □

Though, GPARs have wide applications, they bring several challenges. (1) Conventional support and confidence metrics no longer work for GPARs. (2) Prior techniques cannot be directly applied to discover generalized GPARs. (3) Social graphs are often big and distributively stored, these make mining computation even harder.

Contributions. The paper provides methods to discover GPARs.

(1) We propose generalized GPARs to capture complex social relations among social entities (Sect. 2.2). We define support and confidence metrics for GPARs, decompose the GPARs mining problem into two subproblems, *i.e.,* frequent pattern mining and rules generation, and outline an algorithm for the problem (Sect. 2.3).
(2) We first study the frequent pattern mining problem (Sect. 3.1). We develop a parallel algorithm, that outputs a *DFS code graph* $\mathcal{G}_c$, with nodes corresponding to frequent patterns. The algorithm has desirable performance: it computes support for a node at k-th level in $\mathcal{G}_c$ in $O(|E_f|((k+1)2^{k+1})^{k-1})$ time, where $|E_f|$ is the number of crossing edges. We also study how to generate GPARs with *DFS code graph* $\mathcal{G}_c$ (Sect. 3.2). Given $\mathcal{G}_c = (V_c, E_c)$ and bound η, we then develop an algorithm to produce GPARs with confidence above η in $O(|V_c|(|V_c| + |E_c|))$ time, which is independent of the size of the underlying big graph G.
(3) Using real-life and synthetic graphs, we experimentally verify the performance of our algorithms, and find the following: (a) our mining algorithm scales well with the increase of processors; and (b) they work reasonably well on large graphs (Sect. 4).

Related Work. We categorize related work as follows.

Graph Pattern Mining. The problem has two branches. (1) Algorithms for pattern mining in graph databases are given in [9,10]. (2) Mining techniques over single large graphs are also studied in, *e.g.,* [6]. To improve efficiency, parallel technique is proposed in [13]. Our work differs with [13] in the following: we leverage partial evaluation and asynchronous message passing to identify potential

matches in distributive scenario, moreover frequent patterns are our intermediate results.

GPARs *Mining.* A special kind of GPARs and its mining techniques are introduced in [7], where consequents of the GPARs are defined as pattern graphs with a single edge. Another closer work is about mining GPARs over stream data [11]. Our work differs with them in the semantics, *i.e.,* we are mining generalized GPARs, with antecedent and consequent represented by general graph patterns.

2 Graph Pattern Association Rules

In this section, we introduce graph-pattern association rules.

2.1 Preliminary Concepts

We start with preliminary concepts.

Graph and Subgraph. A *graph* is defined as $G = (V, E, L)$, where (1) V is a set of nodes; (2) $E \subseteq V \times V$ is a set of *undirected* edges ; and (3) each node v in V carries $L(v)$ indicating its label or content *e.g.,* name, job title, as found in social networks. A graph $G' = (V', E', L')$ is a *subgraph* of $G = (V, E, L)$, denoted by $G' \subseteq G$, if $V' \subseteq V$, $E' \subseteq E$, and moreover, for each $v \in V'$, $L'(v) = L(v)$. A *directed* graph is defined similarly, but with each edge (v, v') denoting a directed edge from v to v'.

Pattern and Sub-pattern. A *pattern* Q is a graph (V_p, E_p, f), where V_p and E_p are the set of nodes and edges, respectively; each u_p in V_p has a label $f(u_p)$, specifying search condition, *e.g.,* job title. A pattern $Q' = (V'_p, E'_p, f')$ is *subsumed by* another pattern $Q = (V_p, E_p, f)$, denoted by $Q' \sqsubseteq Q$, if (V'_p, E'_p) is a subgraph of (V_p, E_p), and function f' is a restriction of f.

Isomorphism and Subgraph Isomorphism. An *isomorphism* is a bijective function h from the nodes of Q to the nodes of G, such that (1) for each node $u \in V_p$, $f(u) = L(h(u))$, and (2) (u, u') is an edge in Q if and only if $(h(u), h(u'))$ is an edge in G. A *subgraph isomorphism* [5] is an isomorphism from Q to a subgraph G_s of G. When an isomorphism h from pattern Q to a subgraph G_s of G exists, we say G *matches* Q, and denote G_s as a *match* of Q in G. Abusing notations, we say v in G_s as a *match* of u in Q, when $h(u) = v$.

We denote by $Q(G)$ the set of matches of Q in G. We also denote the image $\mathsf{img}[Q, G]$ of Q in G by a set $\{(u, \mathsf{img}(u)) | u \in E_p\}$, where $\mathsf{img}(u)$, referred to as the image of u in G, consists of distinct nodes v in G as matches of u in Q.

DFS Code and DFS Code Tree. The definitions are introduced in [14]. To make the paper self-contained, we cite them as follows (rephrased).

Given a graph $G = (V, E)$, its DFS tree T_G can be built by performing a depth first search in G from a node. Given T_G, a *DFS code* α of G is an edge sequence $(e_0, e_1, \cdots, e_m)$, that is constructed based on the binary relation $\prec_{E,T}$, such that $e_i \prec_{E,T} e_{i+1}$ for $i \in [0, |E| - 1]$. We refer readers to [14] for more

details about binary relation $\prec_{E,T}$. A graph G can have multiple DFS trees and a set of *DFS codes*. While one can sort them by *DFS lexicographic order*, then the minimum one, denoted by $\mathsf{min}(G)$, can be chosen as the canonical label of G, and graph isomorphism can be determined by comparing $\mathsf{min}(G)$ [14].

A *DFS code tree* $\mathcal{T}_c$ is a directed tree with a single root, where (a) the root is a virtual node, (b) each non-root node, denoted as v_α, corresponds to a *DFS code* α, (c) for a node v_α with *DFS code* $(e_0, \cdots, e_k)$, its child must have a valid *DFS code* in the form of $(e_0, \cdots, e_k, e')$, and (d) the order of the *DFS code* of v_α's siblings satisfies the *DFS lexicographic order*. Similarly, a rooted *DFS code graph* $\mathcal{G}_c$ is a directed graph with a single root as virtual node, and non-root node corresponding to a *DFS code*.

Distributed Data Graphs. A *fragmentation* $\mathcal{F}$ of a graph $G = (V, E, L)$ is $(F_1, \ldots, F_n)$, where each *fragment* F_i is specified by $(V_i \cup F_i.O,\ E_i,\ L_i)$ such that (a) $(V_1, \ldots, V_n)$ is a partition of V, (b) $F_i.O$ is the set of nodes v' such that there exists an edge $e = (v, v')$ in E, $v \in V_i$ and v' is in *another fragment*; we refer to v' as a *virtual node*, e as a *crossing edge* and cE_i as the set of crossing edges; and (c) $(V_i \cup F_i.O, E_i, L_i)$ is a subgraph of G induced by $V_i \cup F_i.O$. In $\mathcal{F}$, we denote $V_f = \bigcup_{i\in[1,n]} F_i.O$ as the set of virtual nodes, E_f as the set of crossing edges, and $|\mathcal{F}|$ as the number of fragments.

We will use the following notations. (1) Pattern Q is *connected* if for each pair of nodes in Q, there exists a path between them. (2) Given a DFS tree, we denote the node being visited lastly via preorder search as the *rightmost node*, then the direct path from root to the rightmost node is named as the *rightmost path*. (3) Given pattern Q, we refer edges that are in the DFS tree as *forward edges*, and remaining edges as *backward edges*. (4) We refer *DFS code*, *DFS code tree* and *DFS code graph* as *code*, *code tree* and *code graph*, respectively, when it is clear from context. (5) Given code α, we refer its corresponding pattern as $Q(\alpha) = (V(\alpha), E(\alpha))$. (6) The *size* $|G|$ of G (resp. $|Q|$ of Q) is $|V| + |E|$ (resp. $|V_p| + |E_p|$), the total number of nodes and edges in G (resp. Q). (7) Given a directed graph G, node v' is a *descendant* of v if there is a directed path from v to v'. (8) Given a directed graph G with a single root v_R, we denote node v as the k-th level node of G, if there exists a path from v_R to v with k edges on the path; and denote the longest path from v_R to a node v in G as the height of G.

2.2 Graph Pattern Association Rules

We now define graph-pattern association rules.

GPARs. A *graph-pattern association rule* (GPAR) R is defined as $Q_l \Rightarrow Q_r$, where Q_l and Q_r (1) are both patterns, and (2) share nodes but have no edge in common. We refer to Q_l and Q_r as the *antecedent* and *consequent* of R, respectively.

The rule states that in a graph G, if there is an isomorphism mapping h_l from Q_l to a subgraph G_1 of G, then there likely exists another mapping h_r from Q_r to another subgraph G_2 of G, such that for each $u \in V_l \cap V_r$, if it is mapped by h_l to v in G_1, then it is also mapped by h_r to the same v in G_2.

We model a GPAR R as *a graph pattern* Q_R by extending Q_l with edge set of Q_r. We consider nontrivial GPARs by requiring that (1) Q_R, Q_l and Q_r are *connected*; and (2) Q_l and Q_r are *nonempty*, *i.e.,* each of them has at least one edge.

2.3 GPARs Mining

We define support and confidence for GPARs, followed by a mining algorithm.

Support. The support of a pattern Q in a single graph G, denoted by $\mathsf{supp}(Q, G)$, indicates the appearance frequency of Q in G. Several *anti-monotonic* support metrics for graph patterns are introduced. Following "minimum image" [6], which preserves *anti-monotonic* property, we define support of Q as $\mathsf{supp}(Q, G) = \mathsf{min}\{|\mathsf{img}(u)| \mid u \in V_p\}$. Then the support of a GPAR R can be defined as $\mathsf{supp}(Q_R, G)$.

Confidence. To find how likely Q_r holds when Q_l holds, we define the *confidence* of a GPAR R in G as $\mathsf{conf}(R, G) = \frac{\mathsf{supp}(Q_R, G)}{\mathsf{supp}(Q_l, G)}$.

Mining Algorithm. Following traditional strategy for association rules mining, we outline an algorithm, denoted by GPARMiner (not shown) for mining GPARs. The algorithm first mines frequent patterns Q_R with support above threshold from graph G, then generates a set of GPARs satisfying confidence threshold with Q_R. In Sect. 3, we will introduce how to mine frequent patterns and generate GPARs.

3 Graph Pattern Association Rules Mining

We first introduce the *frequent pattern mining* (FPM) problem, followed by distributed technique for the problem. We then develop method to generate GPARs.

3.1 Distributed Frequent Pattern Mining Algorithm

The FPM problem can be stated as follows: given a graph G, and support threshold θ, it is to find a set S of frequent patterns Q such that $\mathsf{supp}(Q, G) \geq \theta$ for any Q in S. However, the problem is nontrivial. Its decision problem is verified NP-hard by reduction from the NP-complete subgraph isomorphism problem [5]. Despite the hardness, one can leverage parallel computation to improve mining processing. Motivated by this, we develop a distributive algorithm for the FPM problem.

Theorem 1. *There exists a parallel algorithm for* FPM *problem that computes a code graph $\mathcal{G}_c$ of a fragmented graph $\mathcal{F}$ such that (a) each node in $\mathcal{G}_c$ corresponds to a code representing a frequent pattern, and (b) the support computation for a node at k-th level in $\mathcal{G}_c$ is in $O(|E_f|((k+1)2^{k+1})^{k-1})$ time.*

Proof. We show Theorem 1 by presenting an algorithm as a constructive proof.

Algorithm FPMiner /* executed at the *coordinator* */

Input: Fragmented graph $\mathcal{F} = \{F_1, \cdots, F_n\}$ of data graph G, support threshold θ.
Output: A *code graph* $\mathcal{G}_c$.
1. initialize a *code graph* $\mathcal{G}_c$ rooted at v_R;
2. collect partial results from *workers*; initialize a set QSet;
3. remove α from QSet if $\mathsf{supp}(Q(\alpha), G) < \theta$;
4. **for each** *code* α in QSet **do** $\mathsf{PatExt}(R, \alpha, \theta, \mathcal{G}_c)$;
5. **return** $\mathcal{G}_c$.

Procedure PatExt
Input: α_p, α, θ and $\mathcal{G}_c$.
Output: Updated $\mathcal{G}_c$.
1. **if** $\alpha \neq \mathsf{min}(Q(\alpha))$ **then**
2. connect v_α with its parent v_{α_p} in $\mathcal{G}_c$;
3. **return** ;
4. initialize a new node v_α, connect v_α with its parent v_{α_p} in $\mathcal{G}_c$;
5. generate a set cSet of code, corresponding to a set of candidate patterns;
6. **for each** code α_c in cSet **do**
7. M := M∪ LocalMine $(F_i, \theta, \alpha, \alpha_c)$;
8. **for each** α_c in M **do**
9. compute global support of $Q(\alpha_c)$;
10. **if** $\mathsf{supp}(Q(\alpha_c), G) \geq \theta$ **then** $\mathsf{PatExt}(\alpha, \alpha_c, \theta, \mathcal{G}_c)$;
11. **return** $\mathcal{G}_c$;

Fig. 2. Algorithm FPMiner

The algorithm, denoted as FPMiner and shown in Fig. 2, takes a fragmented graph $\mathcal{F} = (F_1, \ldots F_n)$ and support threshold θ as input, works with a *coordinator* S_c and a set of *working sites* (*a.k.a. workers*) S_i. Before illustrating the algorithm, we first introduce a notion of *partial matches*, and auxiliary structures used by FPMiner.

Partial Matches. Consider that some matches of a pattern may cross over multiple sites, and each *worker* may only have a part of these matches. We refer these match fragments at *workers* as *partial matches*, and associate a unique id with them if they belong to the same match. At each *worker*, we associate a Boolean variable $X_{(u,v)}$ with a virtual node v to indicate whether v is a match of a pattern node u, and send $X_{(u,v)}$, to neighbor sites for local evaluation. When v has not been determined a match of u, $X_{(u,v)}$ is set as unknown; while once v is confirmed a valid (resp. invalid) match of u, $X_{(u,v)}$ is evaluated as true (resp. false).

Auxiliary Structures. The algorithm maintains the following: (a) at the *coordinator*, a *code graph* $\mathcal{G}_c$, in which a node, denoted as v_α, corresponds to a *code* α and the frequency of $Q(\alpha)$; and (b) at each *worker*, indexes $\mathsf{M_F}$, $\mathsf{M_S}$ as hash-table, that map codes corresponding to frequent patterns to their match set and images, respectively.

Algorithm. Below, we describe details of the algorithm.

Initialization. The initialization has following three steps.

(1) The *coordinator* first initializes a *code graph* $\mathcal{G}_c$ with a single node v_R as its root (line 1). It next requests local frequency of patterns from all *workers*.
(2) Upon receiving requests from S_c, all *workers* S_i compute local support for single edge patterns, in parallel as following. (a) Each *worker* S_i constructs a graph G_e with a single edge $e = (v, v')$, and generates its *minimum code* $\mathsf{min}(G_e)$. (b) If $\mathsf{min}(G_e)$ equals to a code α in $\mathsf{M_F}$, S_i (i) checks whether the edge e is a crossing edge or not. If e is a crossing edge, S_i marks v' in G_e as a dummy node, notifies S_j to generate a match of $Q(\alpha)$ and update indexes by sending $X_{(u',v')} = \mathsf{true}$ to S_j, where v' locates; and (ii) updates $\mathsf{M_S}(\alpha)$ by extending $\mathsf{img}(u)$ with $h(u)$ for each pattern node u, where h is the isomorphism mapping from $Q(\alpha)$ to G_e, and expands $\mathsf{M_F}(\alpha)$ with G_e. Note that if a pattern node u is mapped to a dummy node v, v is viewed as a "virtual" match of u, and will not be included in $\mathsf{img}(u)$ at local fragment, instead, v will be included in $\mathsf{M_S}(\alpha)$ at the site where it locates. Finally, S_i generates a set $\mathsf{M} = \{(u, |\mathsf{img}(u)|) | u \in V(\alpha)\}$ for each α in $\mathsf{M_S}$ as local supports, and sends M to the *coordinator*.
(3) The *coordinator* S_c assembles sets M from all *workers*, initializes QSet by summing up all local supports (line 2). It next eliminates *code* α from QSet if $Q(\alpha)$ is not frequent, *i.e.,* frequency is less than threshold θ (line 3), and repeatedly invokes procedure PatExt to mine larger patterns by using frequent single edge patterns (line 4).

Frequent Pattern Mining. Starting from code α that corresponds to a single edge pattern, S_c invokes procedure PatExt to mine frequent patterns. In a nutshell, the mining procedure consists of three stages. Below we describe details of each stage.

(1) *Redundancy Elimination.* A pattern may have numerous *codes*, and each of them may generate different patterns, that might be generated before. Expansion from a code that's generated before may cause unnecessary computation. To avoid this, one can determine whether a code is redundant by applying the rule given by Lemma 1.

Lemma 1. *If the DFS code α of a pattern $Q(\alpha)$ is not minimum, then α must be redundant, and any descendant of v_α on DFS code graph must also be redundant.*

Lemma 1 tells us that one can determine whether code α is redundant by checking whether α is the *minimum code* of the pattern $Q(\alpha)$. With the lemma, PatExt applies the strategy, given in [14], to verify whether a code α is minimum (line 1). The strategy iteratively (a) expands α'; and (b) compares α with α' during expansion, and terminates current iteration as soon as α is already greater than α'.

Note that PatExt uses a *code graph* $\mathcal{G}_c$ to maintain frequent patterns. As opposed to the growth of *code trees* that only includes a new edge between a pair of nodes that corresponding to *minimum codes*, given code α and its parent

Procedure LocalMine /* executed at each site S_i in parallel */

Input: Fragment F_i, frequency threshold θ, α, α_c.
Output: A set M.
1. **for each** $G_{Q(\alpha)}$ (with id=id) in $\mathsf{M_F}(\alpha)$ as a match of $Q(\alpha)$ **do**
2. generate $G_{Q(\alpha_c)}$ by extending $G_{Q(\alpha)}$ with $e_1 = (v_1, v_1')$;
3. **if** $G_{Q(\alpha_c)}$ is a match of $Q(\alpha_c)$ **then**
4. assign id_c to $G_{Q(\alpha_c)}$; update $G_{Q(\alpha_c)}$, $\mathsf{M_S}(\alpha_c)$, $\mathsf{M_F}(\alpha_c)$;
5. **for each** undetermined virtual node v_v (locating at S_j) in $G_{Q(\alpha_c)}$ **do**
6. set corresponding variable associated with v_v as true;
7. invoke $\mathsf{EvalT}(id, \alpha, \alpha_c)$ at S_j;
8. **if** $G_{Q(\alpha_c)}$ is a possible match of $Q(\alpha_c)$ and v_1' is at site S_j **then**
9. invoke $\mathsf{EvalU}(id, u_1, v_1, u_2, v_2)$ at S_j;
10. **if** $X_{(u_1', v_1')}$=true **then**
11. update $G_{Q(\alpha_c)}$, $\mathsf{M_S}(\alpha_c)$, $\mathsf{M_F}(\alpha_c)$;
12. **return** $\mathsf{M} = \{(\alpha_c, \langle u, |\mathsf{S}(u)|\rangle) | u \in V(\alpha_c), \alpha_c \in \mathsf{M_S}\}$;

Fig. 3. Procedure LocalMine

code α_p, if α is redundant, $\mathcal{G}_c$ will also be expanded by inserting an edge from node v_{α_p} to node $v_{\alpha'}$, where α' is the *minimum code* of $Q(\alpha)$, even though pattern extension will no longer proceed (lines 2–3). If otherwise α is a minimum code, PatExt expands $\mathcal{G}_c$ with node v_α and edge (v_{α_p}, v_α) (line 4), and starts next round pattern extension (Fig. 3).

(2) *Candidate Generation.* To avoid generate duplicate patterns following naive pattern extension strategy, below rule is applied by PatExt. Specifically, given *code* α, (a) the new edge to be inserted in $Q(\alpha)$, either connects nodes on the rightmost path of DFS tree of $Q(\alpha)$ as backward edges, or is grown from rightmost node of $Q(\alpha)$ as forward edges; and (b) any node label of G, whose appearance frequency is above support threshold θ can be used to label new node in the updated pattern (line 5).

Example 2. Given a pattern $Q_3 = (\mathsf{DBA}, \mathsf{BA})$, three candidates $Q_{3.1}$, $Q_{3.2}$ and $Q_{3.3}$, shown in Fig. 1(d), are generated by expanding Q_3 with forward edges (marked with dotted lines), while candidate $Q_{3.1.4}$ extends $Q_{3.1}$ with a backward edge (marked with dotted lines). Moreover, the *code* of $Q_{3.1.2}$: $\{(\mathsf{BA}, \mathsf{DBA}), (\mathsf{DBA}, \mathsf{PRG}), (\mathsf{DBA}, \mathsf{ST})\}$ is already a minimum code. When candidate $Q_{3.2.2}$ is generated, it is considered a duplicate pattern as its *minimum code* is the same as that of $Q_{3.1.2}$.

(3) *Support Computation.* The support of candidate patterns is computed in parallel, with the following three steps.

Step I: Given a set cSet of *codes*, the *coordinator* iteratively sends a candidate code α_c to *workers* to request local support of $Q(\alpha_c)$ (lines 6–7 of PatExt).

Step II: Upon receiving α_c, all *workers* compute matches and *image* of pattern $Q(\alpha_c)$ with procedure LocalMine, in parallel. Specifically, each *worker* S_i identifies the edge $e_i = (u_1, u_1')$ that is in $Q(\alpha_c)$ but not in $Q(\alpha)$, and generates a candidate match $G_{Q(\alpha_c)}$ by extending $G_{Q(\alpha)}$ with an edge $e_1 = (v_1, v_1')$, whose endpoints have the same node labels as that of edge e_i, for each match $G_{Q(\alpha)}$ of $Q(\alpha)$ (line 2). Note that the set of matches $G_{Q(\alpha)}$ are generated in last round computation and are stored in $\mathsf{M_F}$. If $G_{Q(\alpha_c)}$ is verified a valid match of $Q(\alpha_c)$ (line 3), *worker* S_i (a) assigns an id id_c to $G_{Q(\alpha_c)}$; (b) marks v_1' in $G_{Q(\alpha_c)}$ as a "dummy" node, if v_1' is a virtual node; and (c) updates indexes $\mathsf{M_S}(\alpha_c)$ and $\mathsf{M_F}(\alpha_c)$ (line 4). It next propagates"truth" value of each virtual node in $G_{Q(\alpha_c)}$ via procedure EvalT (lines 5–7). While if $G_{Q(\alpha_c)}$ can not be determined a match of $Q(\alpha_c)$, which indicates that $e_i = (u_1, u_1')$ and $e_1 = (v_1, v_1')$ are backward edge and crossing edge, respectively, then EvalU (not shown) is invoked at S_j, where v_1' locates, to verify whether $G_{Q(\alpha_c)}$ is a true match or not (lines 8–9). After verification, if $X_{(u_1,v_1)} = \mathsf{true}$, representing that v_1 is a valid match of u_1, is returned, EvalU updates $G_{Q(\alpha_c)}$, $\mathsf{M_S}(\alpha_c)$ and $\mathsf{M_F}(\alpha_c)$ in the same way as it does before (lines 10–11). Lastly, LocalMine sends the set $\mathsf{M} = \{(\alpha_c, \langle u, |\mathsf{S}(u)|\rangle) | u \in V(\alpha_c), \alpha_c \in \mathsf{M_S}\}$ to the *coordinator* for support computation (line 12).

Procedure EvalT (not shown) propagates truth value of variables as follows. Upon receiving id, α and α_c, it checks whether there exists a partial match with id=id in local index $\mathsf{M_F}(\alpha)$, if there does not exist such a partial match, then a new match $G_{Q(\alpha_c)}$ of $Q(\alpha_c)$ is generated, where all the nodes in $G_{Q(\alpha_c)}$ that is not in current fragment are marked as "dummy" nodes; otherwise if the partial match $G_{Q(\alpha)}$ exists, EvalT simply extends $G_{Q(\alpha)}$ with a new edge corresponding to the new pattern edge in $Q(\alpha_c)$. It then updates $\mathsf{M_S}(\alpha_c)$ and $\mathsf{M_F}(\alpha_c)$ by including the new match. For each virtual node v_v in $G_{Q(\alpha_c)}$, if the variable X_{v_v} has not been evaluated, then EvalT sets corresponding variable associated with v_v as true and invokes EvalT at neighbor site where v_v locates for further propagation.

Observe that if pattern $Q(\alpha_c)$ is grown from $Q(\alpha)$ with a backward edge (u_1, u_1'), then any candidate $G_{Q(\alpha)}$ should be expanded with the edge $e_b = (h(u_1), h(u_1'))$, where h is the isomorphism mapping from $Q(\alpha)$ to $G_{Q(\alpha)}$. When the edge e_b is a crossing edge connecting v_1 and v_1', $G_{Q(\alpha_c)}$ may not be determined a valid match of $Q(\alpha_c)$ with local information, then data needs to be shipped to neighbor sites for further verification. In light of this, procedure EvalU is invoked at site where virtual node v_1' locates to check whether there exists a partial match with id=id, and contains v_1' as a match of u_1'. If so, it next invokes procedure EvalT to update indexes and propagate truth variables.

Example 3. After receiving a candidate pattern $Q_{3.1}$ from S_c, the *worker* S_2 computes its local frequency as following. S_2 first extends matches of Q_3 and generates two matches $G_{3.1}^1$ and $G_{3.1}^2$ with node set $\{\mathsf{Nancy}, \mathsf{Fred}, \mathsf{Bill}\}$ and $\{\mathsf{Nancy}, \mathsf{Fred}, \mathsf{Tim}\}$, respectively, and updates $\mathsf{M_S}(\alpha_c)$) and $\mathsf{M_F}(\alpha_c)$ by including $G_{3.1}^1$ and $G_{3.1}^2$, respectively. As $G_{3.1}^2$ has a virtual node Tim as a match of PRG, S_2 evaluates $X_{(\mathsf{PRG},\mathsf{Tim})}$ as true, marks Tim in $G_{3.1}^2$ as dummy node, sends

a tuple with $X_{(\mathsf{PRG},\mathsf{Tim})} = \mathsf{true}$ to S_3, and set $\mathsf{M} = \{(\mathsf{BA}, 1), (\mathsf{DBA}, 1), (\mathsf{PRG}, 1)\}$ to the *coordinator*. After receiving tuple from S_2, *worker* S_3 initializes a new match of $Q_{3.1}$ with node set $\{*, \mathsf{Fred}*, \mathsf{Tim}\}$. □

Step III: At the *coordinator*, procedure PatExt receives sets M from all *workers*, assembles them, computes global support for each candidate pattern $Q(\alpha_c)$, and invokes itself to further mine patterns with $Q(\alpha_c)$ that is verified frequent. When all codes are processed, PatExt returns *code graph* $\mathcal{G}_c$ as the result (lines 9–11 of PatExt).

Result Collection. When all the codes corresponding to single edge patterns are processed, the *code graph* $\mathcal{G}_c$ is built up. The *coordinator* then returns $\mathcal{G}_c$ as final result since each node on $\mathcal{G}_c$ corresponds to a frequent pattern (line 5 of FPMiner).

Analyses. To analyze the performance of the algorithm, we denote candidate patterns corresponding to k-th level nodes of *code graph* $\mathcal{G}_c$ as $Q_k = (V_k, E_k)$. Then $|E_k|$ trivially equals to k. One may verify the following to see the complexity.

Complexity. When computing support for candidate Q_k, the number of matches that need to be expanded is bounded by $O(2^{|V_k|})$. For each crossing edge (v_1, v_1') marked as unknown at S_i, it takes neighbor site $S_j O(2^{|V_k|}|V_k| + |V_k||F_j.O|)$ time to verify whether v_1' is a match of u_1'. Moreover, the verification request will be propagated within $|E_k| - 1$ steps from S_i. Hence, it takes $O(2^{|V_k|} + |F_i.O| + |cE_i|(2^{|V_k|}|V_k| + |V_k||F_i.O|)^{|E_k|-1})$ time, which is bounded by $O(|E_f|((k+1)2^{k+1})^{k-1})$ time, to compute support for Q_k.

This completes the proof of Theorem 1. □

3.2 Rule Generation

Below we present an algorithm to generate GPARs with *code graph* $\mathcal{G}_c$.

Algorithm. The algorithm, denoted as RuleGen (not shown), takes as input a *code graph* $\mathcal{G}_c = (V_c, E_c)$ and confidence bound η. It first initializes empty set S and queue q. It next repeatedly traverses $\mathcal{G}_c$ from each node v_α by reverse breadth first search. For each ancestor $v_{\alpha''}$ of v_α encountered during the traversal, it generates another pattern Q_r by excluding edges of $Q(\alpha'')$ from Q_α. If Q_r is connected and the confidence $\frac{\mathsf{supp}(Q(\alpha),G)}{\mathsf{supp}(Q(\alpha''),G)} \geq \eta$, a new GPAR $Q(\alpha'') \Rightarrow Q_r$ is generated and included in set S. Lastly, RuleGen returns the set S as the final result after all the nodes are processed.

Analyses. To see the complexity of RuleGen, observe that the number of nodes in $\mathcal{G}_c$ is $|V_c|$, and each round of breadth first search takes at most $|V_c| + |E_c|$ time, hence the algorithm RuleGen is in $O(|V_c|(|V_c| + |E_c|))$ time. □

4 Experimental Study

Using real-life and synthetic data, we conducted following experiments to evaluate (1) the scalability of algorithm FPMiner, and (2) the efficiency of algorithm RuleGen.

Experimental Setting. We used three real-life graphs: (a) *Pokec* [2], a social network with 1.63 million nodes taking 269 different node types, and 30.6 million edges; (b) *Google+* [8], a social graph with 4 million entities of 5 types and 53.5 million links; and (c) *Web* [3], a snapshot of Web graph with 12.1 million Web pages labeled by its domain or country using a label set $\mathcal{L}$ with $|\mathcal{L}| = 100$ and 103.6 million links. We designed a generator to produce synthetic graphs $G = (V, E, L)$, controlled by the numbers of nodes $|V|$ and edges $|E|$, where L is taken from an alphabet of $1K$ labels.

Algorithms. We implemented the following, all in Java. (1) Algorithm FPMiner, compared with (a) $\mathsf{GRAMI}_{\mathsf{ND}}$, which is a naive distributed algorithm, that ships all fragments to the *coordinator*, and applies centralized FPM tool GRAMI [1]; and (b) $\mathsf{GRAMI}_{\mathsf{D}}$, another distributed FPM algorithm. $\mathsf{GRAMI}_{\mathsf{D}}$ first computes support of single edge patterns in the same way as FPMiner does, and broadcasts frequent patterns to each *worker*. All *workers* then invoke GRAMI [6] to compute local supports parallelly, and ship both local supports and those matches with *virtual nodes* to the *coordinator*. The *coordinator* finally assembles results and identifies frequent patterns. (2) Algorithm RuleGen.

Graph Fragmentation and Distribution. We used the algorithm of [12] to partition graph G into n fragments, and distributed them to n sites ($n \in [1, 20]$). Each site is powered by 8 cores Intel(R) Xeon(R) 2.00 GHz CPU with 128 GB of memory and 1 TB hard disk, using Debian Linux 3.2.04 system. Each experiment was run 5 times and the average is reported.

Experimental Results. We next report our findings.

Exp-1: Scalability of FPMiner. In this set of experiments, we evaluated the scalability of FPMiner by varying the number of fragments n. We use *logarithmic scale* for the y-axis in Fig. 4(a)–(b). We started the tests with three real-life datasets.

Varying n. Fixing $\theta = 3K$, $1K$, and $4K$ for *Pokec*, *Google+* and *Web*, respectively, we varied n from 4 to 20 and evaluated efficiency of FPMiner. Figure 4(a), (b) and (c) report the results of FPMiner on *Pokec*, *Google+* and *Web*, respectively, which tell us the following. Algorithm FPMiner allows a high degree of parallelism: The more sites are available, the less time it takes. It is, on average, 4.1, 3.3, and 2.9 times faster when n increases from 4 to 20 on *Google+*, *Web* and *Pokec*, respectively, and scales best among three algorithms. Moreover, it takes 749 s for FPMiner over *Web*, with 20 sites.

Varying θ. Fixing $n = 4$, we varied support θ from $2K$ to $4K$ in $0.5K$ increments, $0.6K$ to $1.0K$ in $0.1K$ increments, and $3K$ to $5K$ in $0.5K$ increments on *Pokec*,

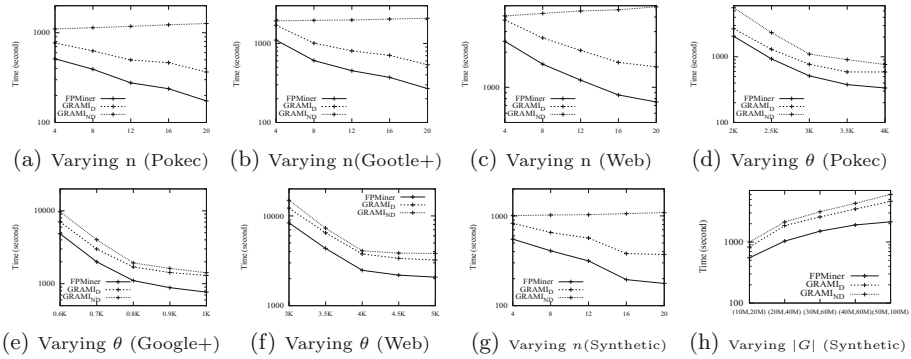


Fig. 4. Performance evaluation

Google+, and *Web*, respectively. Figure 4(d), (e) and (f) tell us the following. (1) All algorithms take longer with small θ, because more candidate patterns and their matches need to be verified. (2) FPMiner outperforms $\mathsf{GRAMI}_\mathsf{D}$ in all cases, and is less sensitive to the increment of θ. This is because FPMiner maximizes parallelism during support computation, hence is most efficient; $\mathsf{GRAMI}_\mathsf{D}$ assembles matches that cross multiple sites, and verifies support with costly centralized method; and $\mathsf{GRAMI}_\mathsf{ND}$ is essentially a centralized algorithm, and has worst performance, which is as expected.

Varying n (Synthetic). Fixing $|G| = (10M, 20M)$, and $\theta = 4K$, we varied n from 4 to 20. The results are shown in Fig. 4(g), and consistent with Fig. 4(a), (b) and (c). In particular, FPMiner takes 177 s with 20 processors on synthetic graph G.

Varying θ (Synthetic). Fixing $n = 4$, $\theta = 4K$, we varied $|G|$ from $(10M, 20M)$ to (50M, 100M) with $10M$ and $20M$ increments on $|V|$ and $|E|$, respectively. As shown in Fig. 4(h), (1) all three algorithms take longer on larger graphs; (2) FPMiner is less sensitive to $|G|$ than others, since when graphs get larger, the increment of its computational time grows slower than other algorithms; and (3) FPMiner outperforms $\mathsf{GRAMI}_\mathsf{D}$ and $\mathsf{GRAMI}_\mathsf{ND}$ by 1.6 and 2 times, respectively, on average, which is consistent with Fig. 4(d), (e) and (f).

Exp-2: Performance of RuleGen. We evaluated efficiency of RuleGen in this test.

Efficiency. Fixing confidence threshold $\eta = 0.6$, we varied support θ from $2K$ to $4K$ in $0.5K$ increments, $0.6K$ to $1.0K$ in $0.1K$ increments, and $3K$ to $5K$ in $0.5K$ increments on *Pokec*, *Google+*, and *Web*, respectively, and evaluated efficiency of RuleGen. We find that (1) RuleGen is very efficient, taking only 754 ms on *Web* when $\theta = 3K$; (2) RuleGen spends more time when θ gets smaller, as the smaller θ is, the bigger *code graph* $\mathcal{G}_c$ is, hence more time is needed for the traversal.

5 Conclusion

We have proposed generalized graph-pattern association rules (GPARs) and viable support, confidence measures, and shown that GPARs can be used to identify association rules among entities in social graphs. We have provided techniques for GPARs mining. Our experimental study has verified the performance of the algorithms. We contend that GPARs yield a promising tool for social network analysis.

Acknowledgments. This work is supported by NSFC 71490722, and Fundamental Research Funds for the Central Universities, China.

References

1. GraMi. https://github.com/ehab-abdelhamid/GraMi
2. Pokec social network. http://snap.stanford.edu/data/soc-pokec.html
3. Web graph. http://law.di.unimi.it/datasets.php
4. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. SIGMOD Rec. **22**(2), 207–216 (1993)
5. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. TPAMI **26**(10), 1367–1372 (2004)
6. Elseidy, M., Abdelhamid, E., Skiadopoulos, S., Kalnis, P.: GRAMI: frequent subgraph and pattern mining in a single large graph. PVLDB **7**(7), 517–528 (2014)
7. Fan, W., Wang, X., Wu, Y., Xu, J.: Association rules with graph patterns. PVLDB **8**, 1502–1513 (2015)
8. Gong, N.Z., et al.: Evolution of social-attribute networks: measurements, modeling, and implications using google+. In: IMC (2012)
9. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: SIGKDD (2004)
10. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45372-5_2
11. Namaki, M.H., Wu, Y., Song, Q., Lin, P., Ge, T.: Discovering temporal graph association rules. In: CIKM (2017)
12. Rahimian, F., Payberah, A.H., Girdzijauskas, S., Jelasity, M., Haridi, S.: JA-BE-JA: a distributed algorithm for balanced graph partitioning. In: SASO (2013)
13. Talukder, N., Zaki, M.J.: A distributed approach for graph mining in massive networks. Data Min. Knowl. Discov. **30**(5), 1024–1052 (2016)
14. Yan, X., Han, J.: GSPAN: graph-based substructure pattern mining. In: ICDM (2002)

Database System Architecture and Performance

Cost Effective Load-Balancing Approach for Range-Partitioned Main-Memory Resident Data

Djahida Belayadi[1], Khaled-Walid Hidouci[1], Ladjel Bellatreche[2(✉)], and Carlos Ordonez[3]

[1] LCSI Laboratory, Ecole Nationale Superieure d'Informatique, Algiers, Algeria
[2] LIAS/ISAE-ENSMA, 86960 Futuroscope, France
ladjel.bellatreche@ensma.fr
[3] University of Houston, Houston, USA

Abstract. Due to the availability of larger RAM capacity, there is a new trend bringing parallel main memory database systems with higher performance, compared to traditional DBMSs. In parallel database systems the most critical aspect is data partitioning, which significantly impacts query processing time. Specifically, unbalanced data partitioning introduces data skew, which ends up decreasing query performance if not managed. In this work, we focus on optimizing range queries, widely used in P2P, decision support systems and spatio-temporal databases. We improve the communication complexity of the state-of-the-art previous algorithm based on skip graphs, which required O(log p) messages between 2 nodes to rebalance load, resulting in a high complexity O(p log p) to rebalance load on the p nodes. With such high cost in mind, we propose to create a global view of data distribution among all processing nodes and database clients. Our main contribution is the Approximate Partitioning Vector ($\mathcal{APV}$), which provides a global approximate view of data distribution to both processing nodes and database clients. A new data balancing algorithm, following a ring topology, reduces communication to 2 messages per node pair, resulting in O(1) communication complexity per node pair and O(p) globally among the p nodes. Experiments analyze the tradeoff between adjusting load balance and query performance.

Keywords: Data skew · Load-balancing · Parallel database
Range-partitioning · Main memory databases · P2P databases
Parallel query processing

1 Introduction

Range-partitioning maps tuples to partitions according to a partitioning key. This scheme is the most common type of partitioning [1] and is often used

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 239–249, 2018.
https://doi.org/10.1007/978-3-319-98812-2_20

with times (hour/minute). A key requirement in such systems is that the data has to be uniformly partitioned on all nodes. This requirement is challenging enforcing when the input data is skewed. *Data skew* is a well-known concern in range-partitioning where only few partitions (nodes) may be more loaded than others. In that case, data migration approaches are an appealing solution. The out-of-range data has to be moved from the over-loaded area to under-loaded one in order to satisfy the storage balance requirement. Data movement must be accompanied by a change in the partition statistics (partition boundaries, neighbors loads, and the position of the most and least loaded node, etc.). All nodes/clients of the system must be aware of these changes in order to decide whether to call the balancing algorithm and address the right node. One of the dominant measures that we want to optimize in such a situation is communication cost. The focus is on the solutions that reduce the cost of maintaining data distribution information.

Ganesan et al. [2] proposed an on-line load-balancing algorithm on a linearly ordered nodes. Their algorithm called AdjustLoad guarantees a low imbalance ratio between the maximum and the minimum load among nodes. Although, the AdjustLoad algorithm is easy to state, each balancing operation may require global load information, that may be expensive in term of operations costs. Their algorithm uses a data structure called skip graph [3] to maintain load information and ensure efficient range queries. Each invoked balancing operation may require global information with a cost of O(log p) messages. Moreover, a change of partition boundaries between neighbors in load-balancing will necessitate a change in the two skip graphs used.

In this paper, we improve the Ganesan et al. work [2] by reducing the cost of maintaining partition statistics. We propose an on-line balancing technique of range-partitioned data with approximate information. Our algorithm uses the same primitive operations, NbrAdjust and Reorder as in Ganesan et al. The primitive NbrAdjust transfers the surplus of data from the current node to one of its neighbors. The primitive Reorder changes the nodes order to achieve the storage balancing requirements. As a result, the partition boundaries change as well as the data size of each partition. However, our algorithm is not based on skip graphs to maintain partition statistics. The key point of our contribution is the Approximate Partitioning Vector ($\mathcal{APV}$), where, both nodes and clients have approximate information on data distribution statistics. Each entry $APV[i]$ in this vector, is an estimate of partition boundaries and data size related to node N_i. After a balancing operation, the participating nodes may change their own boundaries. These nodes do not need to inform the other ones by these changes. The clients use their $\mathcal{APV}$ to direct the queries. As a result, clients may address the wrong node when their $\mathcal{APV}$ are outdated. Nevertheless, whenever an interaction happens between two peers (node or client), they exchange their $\mathcal{APV}$ in order to correct each other. Our solution outperforms the state-of-the-art methods in terms of communication cost. There is no additional cost for maintaining load statistic as in Ganesan et al.

This paper is organized as follows: in Sect. 2, we present the load-balancing approach of Ganesan et al. We describe the system model in Sect. 3. In Sect. 4, we present our approach. We experimentally evaluate it in Sect. 5. Finally, we discuss the related work in Sect. 6.

2 Load-Balancing Solution of Ganesan et al. [2]

Ganesan et al.'s work is believed to be representative of the prior art. It suggested an inspiring algorithm for reducing data skew. It guarantees a small imbalance ratio σ between the largest load and the smallest one. This ratio is always bounded by a small constant which is 4.24. The algorithm uses two operations: (1) NbrAdjut, where the node N_i transfers its surplus of data to one of its neighbors (N_{i+1} or N_{i-1}), (2) Reorder: the least loaded node (N_r) among all the nodes transfers its entire content to one of its neighbors and change its logical position to share data with the node performing the load-balancing algorithm.

In both operations, the over-loaded node requires non-local information (neighbors loads, the position of the most and least loaded node). A given node attempts to shed its load whenever its load increases by a factor δ. For some constant c, Ganesan et al. define a sequence of thresholds $T_i = c\delta^i$, for all $i \geq 1$. The node N_i attempts to trigger the AdjustLoad procedure whenever its load $L(N_i)$ is greater than its threshold T_i.

Ganesan et al. use two skip graphs. Skip graphs are circular linked lists [4], in which every node has $log(n)$ pointers. Routing between two nodes needs $O(log n)$ messages. The first skip graph is used to get neighbors loads (one message) and to route range queries to the appropriate node. The second skip graph is used to get the positions of the most and least loaded node in the system ($O(log n)$ messages for locality plus costs of updating the two skip graphs). The main disadvantage of this work is the costly need to maintain the load-balancing information.

3 System Architecture

In this section, we define a simple abstraction of a parallel database and make some assumptions:

- $N = \{N_1, N_2, \ldots, N_p\}$, a set of p nodes. We consider a relation (or a data set) divided into p range-partitions on the basis of a key attribute, with boundaries $R_0 \leq R_1 \leq \ldots \leq R_p$. The node N_i manages a range $[R_{i-1}, R_i]$. We consider that the nodes are ordered by their ranges, this ordering defines left and right relationships between them. The need to preserve the order between the nodes requires a communication only between the neighboring nodes. Note that the data is In-memory resident and it is organized row wise. Each node N_j has its own Approximate partitioning vector APV_{n_j}. An entry $APV_{n_j}[i]$ in this vector (for i different from j), is an estimate of partition boundaries and data size related the node N_i.

- $C = \{C_1, C_2, \ldots, C_m\}$, a set of m clients performing insert, delete or range queries. The clients may join or leave the system at any time. Each client C_j has its own approximate partitioning vector APV_{c_j}. An entry $APV_{c_j}[i]$ in this vector, is an estimate of partition boundaries and data size related to the node N_i. Clients use their approximate partitioning vector APV_c to find the right nodes for insert, delete or search operations.
- We assume that each node N_i sets a local load threshold. When the load goes outside this limit, the node performs a load-balancing operation with its neighbors. This operation updates the partition boundaries of the participating nodes. Our load-balancing algorithm is invoked on a node at which the insert or delete is occurring or a node receiving data from its neighbors. At this stage, we assume that no central site is used to direct queries. We also ignore concurrency control issues and consider only the serial schedule of inserts and deletes, interleaved with the executions of the load-balancing algorithm.

4 Our Approach

The main feature of our approach is the $\mathcal{APV}$ concept, where each node or client has an approximate knowledge about the partition statistics. Based on this knowledge, the node performs a load-balancing whenever its load passes a local threshold. It invokes the NbrAdjust procedure that transfers the out-of-range data and its vector towards its neighbors if it is possible, otherwise, it performs the Reorder procedure. The neighbor, after having received the data, performs the load-balancing algorithm and eventually updates its vector. The process is repeated at each node receiving the data until the whole system is balanced.

4.1 Node/Client Approximate Partitioning Vector

Consider that the partition statistics of a node N_j are encoded in a vector $APV_{n_j}[1, p]$. Each element $APV_{n_j}[i]$ of this vector is a triplet, it stores an estimate of the node N_i information. Node information is mainly the upper boundary ($APV_{n_j}[i].Upper\_Bound$), the local data size ($APV_{n_j}[i].Load$) and the last updating time ($APV_{n_j}[i].Last\_Update$). The last field is used to indicate the time when the entry $APV_{n_j}[i]$ was last updated. Node vector is updated in case of insert or delete queries, range queries and data migration from the current node to one of its neighbors or vice versa.

The data stored in the nodes is manipulated through the insert, delete and range queries sent by the clients. Consider that the partition statistics of a client C_j are encoded in a vector $APV_{c_j}[1, m]$, where each $APV_{c_j}[i]$ stores the information about the node N_i. Node information is essentially its upper boundary ($APV_{c_j}[i].Upper\_Bound$), data size ($APV_{c_j}[i].Load$) and the last updating time ($APV_{c_j}[i].Last\_update$). Inserting, deleting or searching for a tuple with a given key k are performed as follows:

- The client sends the request to N_i so that: $APV_{c_j}[i-1].Upper\_Bound \leq k \leq APVc_j[i].Upper\_Bound$. The local APV_{c_j} vector is also included in the same message.
- A node N_i checks whether the included key k fits its range, if so, it executes the specified request (insert, delete or just point-search), updates eventually its partition statistics and sends a positive acknowledge consisting of its APV_{n_i} to the client.
- If k is outside the node's range, a vector adjustment message (VAM) including a negative acknowledge and the current APV_{n_i} is sent to the client. Another solution may be proposed where the node redirects the request to the appropriate node, this solution may reduce the communication cost but the client vector will be little updated.
- If a client receives a positive acknowledgment from the node, it just updates its vector if it was outdated. Else, if it receives an adjustment message, it updates its vector and repeats the operation until receiving a positive acknowledgment.

4.2 Data Load-Balancing Algorithm

Our load-balancing algorithm uses the two universal load-balancing primitives, Reorder and NbrAdjust as in Ganesan et al.'s work. However, we use the $\mathcal{APV}$ concept to maintain the global load information instead of using the skip graphs. Our algorithm, that we call DataLoadBalancing, is presented bellow (Algorithm 1). A node N_i executes the load-balancing algorithm whenever its load increases beyond a threshold T_i. The algorithm uses its partitioning vector to check if data can be shared with the neighbor with the low load. If the load of one of its neighbors is less than half of N_i's load, then N_i performs the primitive NbrAdjust to average out the load with it. Else, N_i attempts to perform Reorder with the least loaded node in the system. Note that one can avoid the additional launch of the load-balancing algorithm by raising the threshold or more precisely by increasing the factor δ.

5 Experimental Evaluation

In this Section, we present results from our simulation of our approach on a network of 8 nodes and 2 clients. Processing node software and client software were executed on machines with Intel(R) Core(TM) i7-5500U CPU@2.40 GHz and 8GiB of RAM. Algorithms are implemented in C language using the Message Passing Library (Open MPI). In the experiments, we present the algorithm for the insert-only case. This case is simpler to analyze and provides general ideas on how to deal with the general case. It is also of practical interest because in many applications, as in a file sharing, deletions rarely occur. In order to evaluate the new approach in heavy skewed environment, the system is studied under a simulation model that we call $HOTSPOT$. All insert operations are directed to a single hot node. We use a sequence of $5 * 10^4$ frequent insert

Algorithm 1: DataLoadBalancing (N_i, APV_{n_i})

```
Let N_j be the neighbor of N_i with the low load (N_{i+1} or N_{i-1});
if (APV_{n_i}[i].Load/2 ≥ APV_{n_i}[j].Load) then
    //The node will call the NbrAdjust procedure;
    Send the (APV_{n_i}[i].Load − APV_{n_i}[j].Load)/2 tuples to N_j;
    Update N_i's vector
    (APV_{n_i}[i].Load, APV_{n_i}[j].Load, APV_{n_i}[i].Upper_Bound);
    //Re-call the load-balancing procedure again in N_i and N_j;
    DataLoadBalancing (N_i, APV_{n_i}[i]);
    DataLoadBalancing (N_j, APV_{n_i}[j]);
else
    //The node will call the Reorder procedure;
    Find N_r so that: ∀k ∈ [1, p], APV_{n_i}[k].Load ≥ APV_{n_i}[r].Load;
    if (APV_{n_i}[i].Load/4 ≥ APV_{n_i}[r].Load) then
        //N_r is going to change its location to be a N_i's neighbor ;
        Let N_j be the node with the low load between N_{r+1} and N_{r-1}, the two
        neighbors of N_r;
        Send APV_{n_i}[r].Load tuples to N_j;
        N_j changes its position to be N_i's neighbor;
        Send APV_{n_i}[i].Load/2 to N_r;
        Update N_i's vector (APV_{n_i}[i].Load, APV_{n_i}[r].Load, N_i, N_j, and N_r
        upper bounds);
        DataLoadBalancing (N_i, APV_{n_i});
        Rename nodes appropriately after the Reorder;
    else
        The system is balanced;
    end
end
```

operations. The basic performance factors of our load-balancing mechanism are the imbalance ratio, the number of client addressing errors, data movement cost and the number of invocation of DataLoadBalancing algorithm.

5.1 Imbalance Ratio

We measure the imbalance ratio as the ratio between the largest and smallest load after each insert operation. As the system's thresholds are an infinite, increasing geometric sequence, as in Ganesan et al.'s work, we measure the imbalance ratio with three values of the factor $\delta, (\delta = 1.62, \delta = 2, \delta = 4)$. $\delta = 1.62$ is the golden ratio Fig. 1 shows the imbalance ratios (Y-axis) against the number of insert operations (X-axis). When $\delta = 1.62$, the curve shows that the imbalance ratio is always bounded by a constant 6 and it converges to 1.8 after $2 * 10^4$ operations unlike the imbalance ratio of Ganesan et al., that is bounded by 4.24 and converges towards 3.3. The spikes in the curve mean that an invocation of DataLoadBalancing algorithm has been lunched. However, the results of the

3rd experiment $\delta = 4$ are different from the previous ones, the ratio values are larger and converge towards 5.

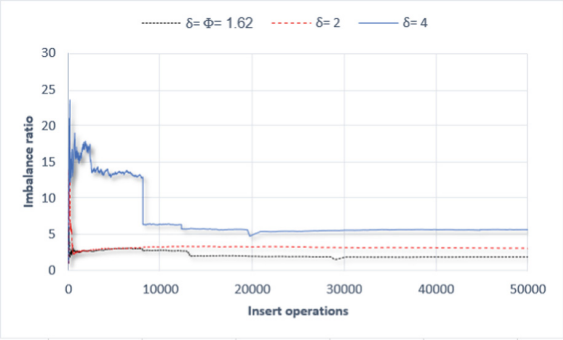


Fig. 1. The imbalance ratio when $\delta = 1.62$ (Golden ratio), $\delta = 2$, and $\delta = 4$ on a cluster of 8 nodes and 2 clients

Table 1. Comparison between the AdjustLoad algorithm of Ganesan et al., and our load DataLoadBalancing algorithm.

Procedure	Largest load (Tuples)	Mean load (Tuples)	Imbalance ratio	Query performance
AdjustLoad	1885	781	2.41	21%
DataLoadBalancing	1492	781	1.91	0
AdjustLoad	2102	1048	2.02	27%
DataLoadBalancing	1568	1048	1.49	0

Simulations were run comparing performance of our load-balancing algorithm and the AdjustLoad procedure of Ganesan et al. [2]. The data from Table 1 represents the comparative load-balancing results of the two procedures. The comparison parameter is the imbalance ratio which is measured here as the ratio between the largest load and the system average load.

When the performance of the system is measured by query response time, it is proportional to the largest node load. The worst-case relative performance of the DataLoadBalancing algorithm versus the AdjustLoad procedure is the ratio of the highest load-balance ratios obtained for the two algorithms, or $1.0 - (1.91/2.41) = 0.21$. One can expect to reduce query response time up to 28% as compared to a system using the AdjustLoad. It is important to mention here that the cost of maintaining the load statistics is at least O(log p), where p is the number of nodes. However, there is no cost needed for our approach.

5.2 Client Adjustment Messages

After a balancing operation, two nodes at least change their partition boundaries due to the data migration, which leads to changing the partitioning vectors of these two nodes. The client with an outdated vector can address a wrong node that has changed its boundaries. In our set of experiments, we were interested in determining how efficiently a client obtains a true view about the nodes. We measure the number of times the client sends a query to a wrong node and hence makes an addressing error and receives a vector adjustment message (VAM). The results showed in Fig. 2 present a rapid increase of addressing errors number in the growing phase (from 1 to 1000 insert operations). This comes back to the fact that there is a frequent invocation of the balancing algorithm, and therefore a frequent change of the partition boundaries.

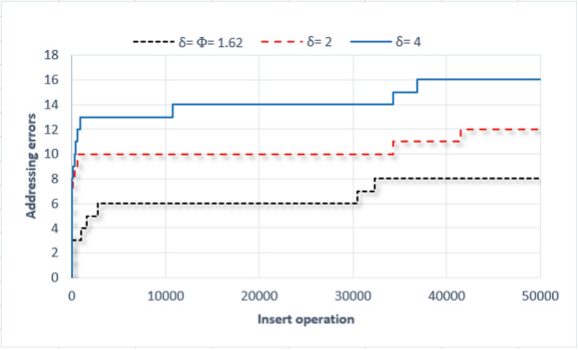


Fig. 2. The number of addressing errors ($\delta = 1.62$ (Golden ratio), $\delta = 2$, and $\delta = 4$) on a cluster of 8 nodes and 2 clients.

5.3 Performance Analysis

For balancing loads among nodes, we are also concerned with the minimization of the movement cost as much as possible. After measuring the imbalance ratio and the client vector adjustment, we next measure the data movement cost. Figure 3(a) plots the cumulative number of tuples migrated by our algorithm (Y-axis) against the number of insert operations (X-axis) during a run with $\delta = 1.62$, $\delta = 2$. We observe that in the growing phase, the number of migrated tuples for different δ values is rising, this is because keeping the system tightly balanced causes a larger number of re-balancing operations. Figure 3(b) plots the data movement cost when $\delta = 4$.

Figure 4 illustrates the number of invocations of our load-balancing algorithm (Y-axis) against the number of insert operations (X-axis) during a run. The observation we make is that the number of invocations of the algorithm increases for the three values of δ during the growing phase. The number of algorithm invocations presented was measured when $\delta = 1.62$, $\delta = 2$ and $\delta = 4$. We

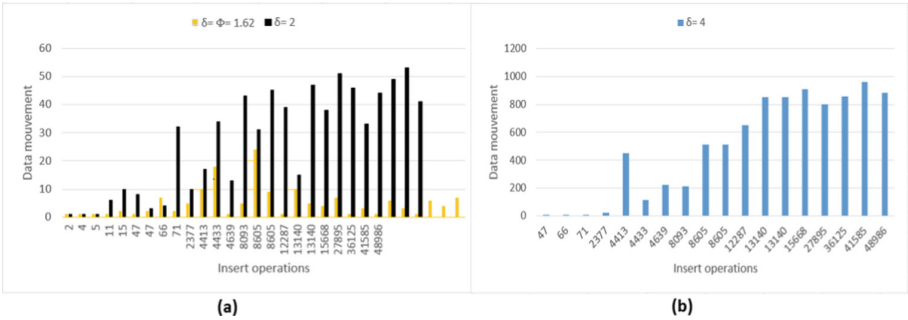


Fig. 3. (a): Data movement cost when $\delta = 1.62$ (Golden ratio), $\delta = 2$. (b): Data movement cost when $\delta = 4$ on a cluster of 8 nodes and 2 clients

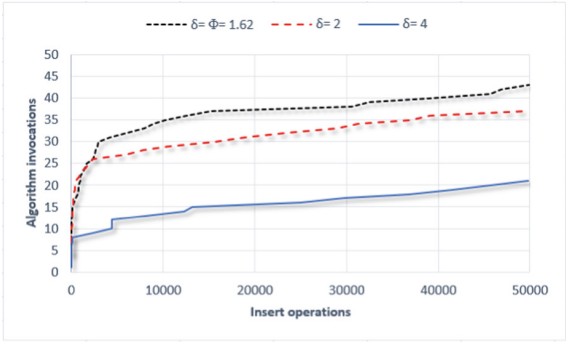


Fig. 4. Number of DataLoadBalancing invocations when $\delta = 1.62$ (Golden ratio), $\delta = 2$ and $\delta = 4$ on a cluster of 8 nodes and 2 clients.

observe that the number of invocations is relatively small when $\delta = 1.62$. This comes back to the fact that we are supporting some imbalance situations.

6 Related Work

In this section, we describe the current researches that relate to ours achieved load-balancing method while supporting range queries.

Peer-to-Peer Network: In P2P networks, a number of recent load-balancing approaches have been proposed [5–8]. Structured P2P networks are an efficient tool for storage and location of data since there is no central server which could become a bottleneck. Many researches have been proposed on search methods in Structured P2P networks.

An improvement of the Ganesan et al.'s work is presented in Chawachat et al. [6]. However, Jakarin et al. use the skip graphs just like Ganesan et al.'s work. The main drawback is the costly maintenance of these data structures. Work in

[9] is an efficient technique to reduce the cost of maintaining partition statistics. Our work is complementary to this one.

Parallel/Distributed Databases: A load-balance operation in a parallel database is performed as a transaction. Work in [10] propose a multi-reorder operation that finds a sequence of multiple adjacent nodes that have a small average load of any such sequence. This technique uses partition statistics which include an estimate of the number of tuples stored on each node for every relation in the database. Based on this information the system data skew can be detected, but at the cost of maintaining partition statistics.

7 Conclusions

The present paper relates to load-balancing in parallel database systems. We proposed an effective on-line data load-balancing algorithm that deals with the problem of skewed data. Our experimental results that we set in our laboratory show that our approach does not need extra cost of maintaining partition statistics as opposed to the cost of efficient solutions from the state-of-art. Our procedure needs a very low overhead (or almost no cost) to locate the data even in the presence of high degree of skew. Although our proposal was presented in the context of balancing storage load, it can be generalized to balance execution load, all that is required is an ability to partition load evenly across two machines. In another future work, we will use the same idea cited here in Spark SQL to avoid data skew. Another major aspects in P2P databases is that some clients may access only some ranges most frequently (popular songs, popular artists). Data can be even partitioned but 90% queries access only 10% of data most of the time. In our next work, we will address this point.

References

1. Cabrera, W., Ordonez, C.: Scalable parallel graph algorithms with matrix-vector multiplication evaluated with queries. Distrib. Parallel Databases **35**(3–4), 335–362 (2017)
2. Ganesan, P., Bawa, M., Garcia-Molina, H.: Online balancing of range-partitioned data with applications to peer-to-peer systems. In: Proceedings of VLDB, Canada, 31 August 2004–3 September 2004 (2004)
3. Aspnes, J., Shah, G.: Skip graphs. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, USA, 12–14 January 2003, pp. 384–393 (2003)
4. Pugh, W.: Skip lists: a probabilistic alternative to balanced trees. Commun. ACM **33**, 668–676 (1990)
5. Felber, P., Kropf, P., Schiller, E., Serbu, S.: Survey on load balancing in peer-to-peer distributed hash tables. IEEE Commun. Surv. Tutor. **16**, 473–492 (2014)
6. Chawachat, J., Fakcharoenphol, J.: A simpler load-balancing algorithm for range-partitioned data in peer-to-peer systems. Networks **66**, 235–249 (2015)

7. Antoine, M., Pellegrino, L., Huet, F., Baude, F.: A generic API for load balancing in structured P2P systems. In: SBAC-PAD Workshop 2014, Paris, France, 22–24 October (2014)
8. Takeda, A., Oide, T., Takahashi, A., Suganuma, T.: Efficient dynamic load balancing for structured P2P network. In: 18th International Conference on Network-Based Information Systems (2015)
9. Belayadi, D., Hidouci, W.: Dynamic range partitioning with asynchronous data balancing. In: 2016 International IEEE Conferences on Smart World Congress, Toulouse, France (2016)
10. Rishel, W.S., Rishel, R.B., Taylor, D.A.: Load balancing in parallel database systems using multi-reordering. US Patent 8,849,749, 30 September 2014

Adaptive Workload-Based Partitioning and Replication for RDF Graphs

Ahmed Al-Ghezi(✉) and Lena Wiese

Institute of Computer Science, University of Göttingen, Göttingen, Germany
{ahmed.al-ghezi,wiese}@cs.uni-goettingen.de

Abstract. Distributed processing of RDF data requires partitioning of big and complex data sets. The partitioning affects the performance of the distributed query processing and the amount of data transfer between the network-connected nodes. Static graph partitioning aims to generate partitions with lowest number of edges in between but suffers high communication cost when a query trespasses a partition's border, because then it requires moving partial results across the network. Workload-aware partitioning is an alternative but faces complex decisions regarding the storage space and the workload orientation. In this paper, we present an adaptive partitioning and replication strategy on three levels. We initialize our system with static partitioning where it collects and analyzes the received workload; then we let it adapt itself with two levels of dynamic replications, besides applying a weighting system to its initial static partitioning to decrease the ratio of border nodes.

1 Introduction

The exploding size of RDF-represented web data led to methods that warehouse and process the data in a distributed system. The first challenge for such systems is to partition the big RDF graph into several fragments, which then need to be assigned wisely to the clustered working nodes. A SPARQL query is represented as a graph on its own with some vertices and/or edges represented as variables. Query execution is the process of finding all the sub-graphs in the RDF-graph that match the query graph. Using several hosts connected in a cluster, a SPARQL query can be received and executed in parallel by all the hosts on their local share of data. Unfortunately, we have an expensive communication cost, whenever the query requires matching sub-graphs located in different hosts. Working towards local query execution and avoiding the communication cost is usually following two directions: (1) a better partitioning strategy, and (2) replication of selected important triples across hosts. Partitioning requires no extra storage space but finding the best strategy often involves solving a difficult problem due to the complex relationships which are embedded in a typical RDF graph. Replication has, in turn, two factors that mainly affect its feasibility: (1) a good selection of graph parts to replicate, and (2) the available storage space which restricts the amount of replicated data. Regarding the first factor, [9]

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 250–258, 2018.
https://doi.org/10.1007/978-3-319-98812-2_21

studied a set of DBpedia's queries: more than 90% of the queries target only 163 frequent sub-graphs. This highlights the impact of the query workload to identify important parts of the RDF graph and parts which should be kept together in a single host. On the other hand, the storage space as the second factor affecting a replication strategy limits the amount of replicated data which any host can handle. However, the amount of storage space in a practical triple-store system is variable; it depends on the size of the data set and of other important data contained in the system like indices and statistics tables. Hence, a partitioning and replication strategy focusing on saving disk space might perform poorly if the system happens to have a lot of free storage space, and vice versa.

To address these challenges, our main contributions in this paper are:

- We present our RDF partitioning and replication approach, which is built on two stages – with respect to the existence and absence of workload – and three levels – with respect to storage space consumption.
- Given that the availability of storage space is variable, we describe the optimized system behaviour with respect to the amount and type of replications, to adapt to the different possible levels of storage space availability.

To the best of our knowledge, this is the first work in a distributed RDF triple store, that considers the adaption of a partitioning and replication layer with both the workload and the storage space availability.

The paper is structured as follows. We discuss related work in Sect. 2, then we introduce the preliminaries and terminologies used in this paper in Sect. 3. We describe the system's initial cold partitioning and replication approach in Sect. 4. Section 5 describes how the system reacts to a collected workload and adapts its storage layer. Finally we state our bench marking report and conclusion in Sect. 6.

2 Related Work

Many recent works deal with general graph partitioning, including works targeting the problem of RDF graph partitioning. METIS [6] is a popular baseline for many works like Huang [5], WARP [4], and TriAD [3] which applies a hash function to assign many METIS partitions to hosts. Other works considere the semantic embedded in the RDF data; Xu [12] apply a Page Rank inspired algorithm to cluster the RDF data. EAGRE [13] identifies an entity concept for the RDF data, which is then used to group and compress the entities which belong to the same RDF classes. Wu et al. [11] follow path partitioning: first identify closed paths in the RDF graph, then assign the paths that share merged vertices to the same partition. Margo and Seltzer [7] perform edge partitioning by converting the graph into an elimination tree and then trying to reduce communication cost while performing partitioning on the resulting tree. Among the workload-based approaches, however WARP [4] and Partout [2] can be considered baselines. Following the work of Partout, Padiya et al. [8] identify properties which are queried together and reflect this when partitioning the data set. Similar to the

min-terms used in Partout and WARP, Peng et al. [9] detect frequent patterns in the workload, measured by the frequency of appearance.

3 Preliminaries

In this section, we state the main definitions (RDF graph, METIS partitioning [6] and the partition's border region) needed throughout this paper.

Definition 1 (RDF Graph). *Let $G = \{V, E, P\}$ be a graph representing the RDF data set. V is a set of all the subjects and objects in the set of RDF triples D; $E \subseteq V \times V$ is a set of directed edges representing all the triples in the data set; P is a set of all the edges' labels in the RDF data, and we denote p_e as the property associated with edge $e \in E$. The RDF data set is then defined as $D = \{(s, p_e, o) \mid \exists e = (s, o) : e \in E \wedge p_e \in P\}$.*

Definition 2 (METIS Partitioning). *We refer to METIS as a function $metis(v)$ which for any $v \in V$ returns the static partition number which v belongs to. We could then define the partition $r_i = \{v \in V \mid metis(v) = i\}$.*

Definition 3 (Border Region). *For a partition r_i, $border(i) = \{v \in r_i \mid \exists (v, v_m) \in E : v_m \notin r_i\}$. The border region with depth δ is defined as the follows: $border(i, \delta) = \{v \in V \mid v \notin r_i, outdepth(v, i) \leq \delta\}$, where the $outdepth(v, i)$ is the distance between any vertex $v \notin r_i$ and the partition border $border(i)$.*

Note that $border(i)$ refers to nodes inside a partition while $border(i, \delta)$ refers to nodes in neighboring partitions. Last, we define the query workload.

Definition 4 (Queries Workload, Query Answer). *A query q is a set of triple patterns $\{t_1, t_2, ..., t_n\}$; each triple pattern $(\grave{s}, \grave{p}, \grave{o}) \in q$ has at least one but not more than two constants, while the rest are variables. This set composes a query graph q_G. The query answer q_a is the set of all sub-graphs in RDF graph G that match the query graph and substitute the corresponding variables. A workload is defined as a set: $Q = \{(q_1, f_1), (q_2, f_2), ..., (q_m, f_m)\}$, where q_i is a SPARQL query, and f_i is the frequency of its appearance in the workload. The workload answer Q_a is the set of the query answers of Q. The length of query q is the maximum distance between any two vertices in its graph q_G.*

4 Cold-Start Partitioning

Without a pre-assumption about the query workload, our system starts by performing static graph partitioning using METIS. This produces n partitions for given n hosts. Each host handles its share of data and can provide space to accept replications. The best area for replications is located at the hosts' border regions with some distance in depth. We differentiate in our system between two types of replication: full and compressed.

4.1 Compressed Replicated Data

In triple stores like RDF3X[1], a dictionary is used to map the textual URIs found in the triples to compressed numerical data which are then stored in different indices. In the context of replication, the later architecture produces two types: *full replication* where the host has full information about the graph data including the dictionary entries, and the *compressed replication* where the host has only the numerical data or even part of it. Given RDF data set represented by graph G which has size T triples, the minimum length of numerical code value used in dictionary is given by: $log_2 \frac{T}{\kappa}$ bits, where κ is the average number of edges per vertex in G. The total size in bytes of the T triples when stored in the numerical form is given by:

$$size(T) = \frac{3T \cdot log_2(\frac{T}{\kappa})}{8} \tag{1}$$

4.2 Initial Cold Replication

At this stage, each host has a given amount of storage space assigned for replication denoted by S, and employs all of this space by replicating triples from its neighbour hosts which are located at $outdepth = 0$ form its border; it then iteratively increases the *outdepth* and replicates the affected triples until the storage space is fully taken. However, at each *outdepth* the host needs to make a decision whether it should replicate the full or compressed data which was explained in Sect. 4.1. The hypothesis here is that the full replication for certain *outdepth* costs more space but performs faster than the corresponding compressed replication. On the other hand, the importance of triples decreases at their *outdepth* increases, since their probability to contribute in queries workload decreases [5].

We now provide cost and benefit functions for each *outdepth* level δ and host i such that each host can make a systematic decision about the type of replication it should perform. The cost function at $outdepth = \delta$ reflects the fraction of the storage space which the host i would pay if this *outdepth* was fully replicated.

$$cost(i, \delta) = \frac{(|border(\delta, i)| - |border(\delta - 1, i)|) \cdot 8 \cdot s_t}{\grave{S}} \tag{2}$$

where s_t is the size of a single compressed triple in Bytes given by Eq. 1, and $\grave{S}$ is the size in Bytes of the currently remaining storage space of the original S input. The factor 8 is estimated practically from [1] as the ratio of full to compressed size. The benefit of host i performing full replication of the triples at $outdepth = \delta$ is related to the fraction of the triples that are expected to participate in the workload, which is inversely proportional with δ:

$$benefit(i, \delta) = \frac{1}{\delta} \cdot \frac{1}{R} \tag{3}$$

[1] https://code.google.com/archive/p/rdf3x/.

The factor R can be greater than 1 when the network performance is relatively poor, or the length of the queries is expected to be small.

For every partition i, we calculate the cost and benefit ratios starting with $\delta = 1$; we make a decision to build full-data replication if the benefit is greater than the corresponding cost, or otherwise consider building compressed replication, or stop if there is no more storage space left.

5 Load-Aware Partitioning and Replication Step

Initial partitioning of the RDF graph with METIS provides solid ground for the system to start executing queries while decreasing the amount of network data transfer between working hosts. Next, our system collects the executed queries in order to perform another partitioning which is based on the workload. In this step, we first update the existing METIS partitioning by performing optimization based on the available workload knowledge, we then support the METIS partitioning by providing efficient replications on multiple levels.

5.1 Weighted METIS

The cold partitioning which took place in Sect. 4, produces METIS partitions that have a minimum number of edges across the partitions, given that all the edges are equally weighted with 1. Since METIS provides the possibility of assigning numerical weights to edges of the input graph, we provide a weighting system that instructs the METIS with the more important edges in order to avoid putting them on the resulting partitions' cut. This weighting system is produced from the collected workload and its results. The key advantage here is that we remove the important vertices from the partition borders, and at the same time we consume no more space, because no new data is added, but it could require moving triples across hosts. The work of [10] compared between different graph partitioning algorithms that try to achieve balanced partitions in terms of the workload, but none of them directly considered the weighting provided by METIS to partition RDF graphs. In contrast, we define the function for each $e \in E$ as $w(e) = J \cdot (f(e) + f(p_e)) + \beta$ where

- $f(e)$ is the frequency of appearance of e in the workload queries answer Q_a,
- $f(p_e)$ is the frequency of appearance of p_e (the property associated with edge e) in the workload query answer,
- β is equal to 0 when the frequency of p_e in the data set is larger than a fixed threshold, or equal to 1 otherwise,
- J is the ratio between the average count of edges per vertex in the RDF graph and the average query frequency in the workload.

5.2 Building Global Query and Fragments Graphs

In this step we again employ the storage space S assigned by each host, by replicating border triples to support the METIS partitions. But we make use

of the collected workload by generating full replication of those border triples which have more priority, and compressed replications for the other lower priority triples. For this we first normalize the workload, generate a global queries-graph, and then describe the algorithm we run to compute the border triples priority.

Workload Normalization

In a query workload Q, we first remove any non-frequent constants (excluding properties) and any variables found in each $q \in Q$ and replace them with single variable Ω. This normalizes the workload and avoids the generation of irrelevant small fragments. The item is considered non-frequent if its frequency is less than a normalization threshold. This threshold was left to the application case in WARP and Partout. We let the system find this threshold by determining the first statistical quartile of the items frequencies, such that we have a fully automated process. Recall from Definition 4 that each q is a set of triple patterns; the normalized workload can be defined as: $T_\Theta = \{(t_{\Theta 1}, f_1), (t_{\Theta 2}, f_2), .., (t_{\Theta k}, f_k)\}$.

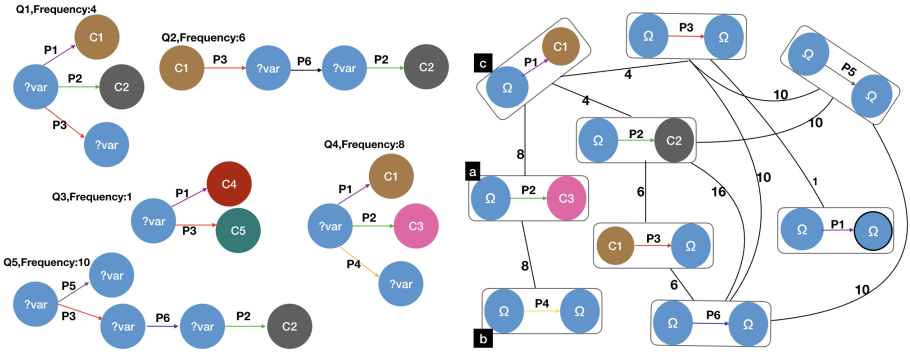


Fig. 1. Example workload (left) and global queries-graph (right)

The Global Queries-Graph

The normalized queries workload is converted into a global queries-graph by modeling each distinct triple pattern as a vertex. Figure 1 shows an example workload and its global queries graph. An edge between two vertices in the graph means that the two triple patterns modeled by the two vertices occur together in one or more queries in the normalized workload. The edge weight represents the total count of this occurrence. We define the global queries-graph as:

Definition 5 (Global Queries-Graph). *For a normalized workload* $T_\Theta = \{(t_{\Theta_1}, f_1), (t_{\Theta_2}, f_2), ..., (t_{\Theta_m}, f_m)\}$, *the global queries-graph is* $G_p = \{V_p, E_p\}$, *where* $V_p = T_\Theta$ *and* $E_p = \{(t_{\Theta_i}, t_{\Theta_j}) \mid \exists (q_k, f_k) \in Q : t_i \in q_k \wedge t_j \in q_k\}$. *Each edge* $(t_{\Theta_i}, t_{\Theta_j})$ *in* G_p *is weighted with the total number of times that the normalized triple pattern* t_{Θ_i} *falls with* t_{Θ_j} *in the same query.*

Algorithm 1: Generate The Workload-based Replication

```
   input : A global queries-graph Q_p = {V_p, E_p}, and RDF graph G = {V, E}
 1 for each host i do
 2     //Create a fragment F_v for each (v, f) ∈ V_p, each F_v is a set of assigned triples:
       F = {F_v | (v, f) ∈ V_p, F_v = {d ∈ D | FragAssign(d) = v}} ;
 3     for δ = 0 to ∞ do
 4         D_t = {(s, p, o) ∈ D | (s ∈ border(i, δ) ∧ o ∈ border(i, δ − 1)) ∨ (o ∈
           border(i, δ) ∧ s ∈ border(i, δ − 1))} ;
 5         if D_t = ∅ then
 6             break;
 7         end
 8         for each d′ ∈ D_t do
 9             for each (t_Θ, f) ∈ V_p do
10                 if d matches t_Θ then
11                     matchImpact ← matchImpact + f ;
12                     q̀ = {(t_Θj, f_j) ∈ V_p | ∃(t_Θ, t_Θj) ∈ E_p};
13                     fragmentMatch ← fragmentMatch + getFragMatch(q̀, t_Θ, d′);
14                     FragNo ← FragAssign(d′)
15                 end
16             end
17             F_FragNo ← F_FragNo ∪ {d′} ;
18             benefit(d′) ←
               benefit(i, δ) · (fragmentMatch / maxRecordedfragmentMatch + matchImpact / maxRecordedMatchImpact);
19             cost(d′) ← cost(i, δ);
20             if benefit(d′) > cost(d′) then
21                 Build full replication of triple d in host i ;
22             else
23                 Build compressed replication of triple d in host i ;
24             end
25         end
26     end
27 end
```

Generating the Workload-Based Replications

The global queries-graph contains the workload-based information which enables the recognition of the border triples that have more probability to participate in future queries. Replicating those triples increases the chance of local query executions. However, such probability is related to the *outdepth* of the border triple and to the engagement level of this border triple with the global queries-graph. The steps we follow in this stage are shown in Algorithm 1. Each host starts from its border and processes the triples at its neighbours which are located at $outdepth = 0$, then iteratively increases the *outdepth*. Those triples are given by set D_t at step 4 in Algorithm 1. We look at each triple $d \in D_t$, and compute two values: The first is $matchImpact$ which is the accumulative frequencies of the normalized triple patterns in G_p that d matches; while the second value is the $fragmentMatch$ which gives an indication about the engagement level of D with G_p. This level is computed by the function $getFragMatch(\grave{q}, t_\Theta, d')$, which computes multiple splits of the triple patterns of $\grave{q}$, given that each split must contain t_Θ, and at least one other $t_{\Theta_j} \in \grave{q}$. The function finds the split l with the maximum number of triple patterns that d' matches, and returns the summation of all edges' weights between t_Θ and other triple patterns in l.

6 Conclusion

In this work we presented a novel adaptive partitioning and replication approach that can highly adapt to different levels of both workload and storage space. We report on a benchmark based on three types of queries and three types of storage levels in [1]. As a test set, we used the YAGO core[2] data set with more than 50M triples. We compare our system with three implementations based on WARP, Partout and Huang.

Acknowledgements. The authors would like to thank Deutscher Akademischer Austauschdienst (DAAD) for providing fund for research on this project.

References

1. Al-Ghezi, A., Wiese, L.: Space-adaptive and workload-aware replication and partitioning for distributed RDF triple stores. In: 29th International Workshop on Database and Expert Systems Applications (DEXA). Springer, Cham (2018)
2. Galárraga, L., Hose, K., Schenkel, R.: Partout: a distributed engine for efficient RDF processing. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 267–268. ACM (2014)
3. Gurajada, S., Seufert, S., Miliaraki, I., Theobald, M.: TriAD: a distributed shared-nothing RDF engine based on asynchronous message passing. In: Proceedings of the ACM International Conference on Management of Data, pp. 289–300. ACM, New York (2014)
4. Hose, K., Schenkel, R.: WARP: workload-aware replication and partitioning for RDF. In: IEEE 29th International Conference on Data Engineering Workshops (ICDEW), pp. 1–6 (2013)
5. Huang, J., Abadi, D.J., Ren, K.: Scalable SPARQL querying of large RDF graphs. Proc. VLDB Endow. **4**(11), 1123–1134 (2011)
6. Karypis, G.: METIS and parMETIS. In: Padua, D. (ed.) Encyclopedia of Parallel Computing, pp. 1117–1124. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-09766-4
7. Margo, D., Seltzer, M.: A scalable distributed graph partitioner. Proc. VLDB Endow. **8**(12), 1478–1489 (2015)
8. Padiya, T., Kanwar, J.J., Bhise, M.: Workload aware hybrid partitioning. In: Proceedings of the 9th Annual ACM India Conference, pp. 51–58. ACM (2016)
9. Peng, P., Chen, L., Zou, L., Zhao, D.: Query workload-based RDF graph fragmentation and allocation. In: EDBT, pp. 377–388 (2016)
10. Shang, Z., Yu, J.X.: Catch the wind: graph workload balancing on cloud. In: Proceedings of the IEEE International Conference on Data Engineering, pp. 553–564. IEEE Computer Society, Washington, DC (2013)
11. Wu, B., Zhou, Y., Yuan, P., Liu, L., Jin, H.: Scalable SPARQL querying using path partitioning. In: 2015 IEEE 31st International Conference on Data Engineering (ICDE), pp. 795–806. IEEE (2015)

[2] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/.

12. Xu, Q., Wang, X., Wang, J., Yang, Y., Feng, Z.: Semantic-aware partitioning on RDF graphs. In: Chen, L., Jensen, C.S., Shahabi, C., Yang, X., Lian, X. (eds.) APWeb-WAIM 2017. LNCS, vol. 10366, pp. 149–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63579-8_12
13. Zhang, X., Chen, L., Tong, Y., Wang, M.: EAGRE: Towards scalable i/o efficient SPARQL query evaluation on the cloud. In: Jensen, C.S., Jermaine, C.M., Zhou, X. (eds.) ICDE, pp. 565–576. IEEE Computer Society (2013)

QUIOW: A Keyword-Based Query Processing Tool for RDF Datasets and Relational Databases

Yenier T. Izquierdo[1,2(✉)], Grettel M. García[1,2], Elisa S. Menendez[1], Marco A. Casanova[1,2], Frederic Dartayre[2], and Carlos H. Levy[2]

[1] Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
{yizquierdo,ggarcia,emenendez,casanova}@inf.puc-rio.br
[2] Instituto TecGraf, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
{fdartayre,levy}@tecgraf.puc-rio.br

Abstract. This paper describes a keyword-based query processing tool, called *QUIOW*, deployed in two distinct environments: RDF datasets with schemas and relational databases. The tool first translates a keyword-based query into an abstract query, and then compiles the abstract query into a concrete (SPARQL or SQL) query such that each result of concrete query is an answer for the keyword-based query. The tool explores the schema to avoid user intervention during the translation process. The paper includes extensive experiments to compare keyword-based query processing in the two environments, using a full version of IMDb – The Internet Movies Database and the Mondial database.

Keywords: Keyword search · SQL · SPARQL · Relational model
RDF

1 Introduction

Database applications that offer keyword search free the users from filling "boxes" with exact data by compiling keyword-based queries to the query language supported, and by ranking the results. In fact, keyword search applications over relational databases have been studied for quite some time. More recently, examples of such applications designed for RDF datasets have emerged.

In this paper, we describe *QUIOW*, a keyword-based query processing tool deployed in two distinct environments: RDF datasets with schemas and relational databases. The tool first translates a keyword-based query into an abstract query, and then compiles the abstract query into a concrete (SPARQL or SQL) query such that each result of concrete query is an answer for the keyword-based query. The tool explores the schema to avoid user intervention during the translation process, and to minimize the number of joins in a query. The implementation of the tool is engineered to work with different RDF stores and relational DBMSs. The current implementation supports Oracle 12c, for both the RDF and relational environments.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 259–269, 2018.
https://doi.org/10.1007/978-3-319-98812-2_22

The paper also covers expensive experiments that use RDF and relational versions of the Mondial database – compiled from geographical Web data sources and a complete variation of IMDb – The Internet Movies Database, which includes descriptions of artists, movies, documentaries, TV series, and even computer games. The experiments with Mondial and IMDb extend the Coffman and Weaver's benchmark [5] and were conducted to fully assess the performance of the tool in the two environments.

The *QUIOW* tool evolved from an earlier implementation [7], which proved efficient, but only worked with RDF datasets and suffered from the problem of maintaining the RDF dataset synchronized with the source relational database. The *QUIOW* tool differs from the earlier tool in four important aspects. First, it includes a backtracking mechanism to generate alternative translations for a keyword-based query, which expands the set of answers returned for the keyword-based query, since the answers now reflect the result of executing several SPARQL (or SQL) queries. Second, the tool supports both the RDF and the relational environments. Third, the negation operator is included in the property filter options. Lastly, the tool is engineered to work with different RDF stores and relational DBMSs.

The contributions of this paper can be summarized as follows. First, the paper describes a keyword-based query processing tool, deployed in two distinct environments: RDF datasets with schemas and relational databases. This is the first tool with these characteristics to be described in the literature, to the best of our knowledge. Second, it includes extensive experiments using Mondial and IMDb to help assess the performance of the tool and compare keyword-based query processing in two environments.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 introduces the keyword translation algorithm the *QUIOW* tool uses. Section 4 describes extensive experiments to compare keyword-based query processing over RDF datasets and relational databases and to assess the performance of the tool. Finally, Sect. 5 presents the conclusions and future work.

2 Related Work

Recent surveys of keyword-based query processing tools over relational databases and RDF datasets can be found in [3, 16]. Early relational keyword-based query processing tools [1, 2, 10, 11, 14] explored the foreign/primary keys declared in the relational schema to compile a keyword-based query into an SQL query with a minimal set of join clauses, based on the notion of candidate networks (CNs). This approach was also adopted in recent tools [4, 13]. In particular, *QUEST* [4] explores the structure of the conceptual schema to synthesize an SQL query, based on a Steiner tree that captures a minimum set of joins. Tastier [13] included the user in the keyword-based query processing loop and provided context-sensitive auto-completion of keyword queries, via specialized data structures that index the database. Coffman and Weaver [5] presented a qualitative evaluation of several relational keyword-based query processing tools, using a benchmark, which consists of a simplified version of IMDb, a subset of

Wikipedia, and a subset of the Mondial dataset, and a set of 50 keyword queries for each database.

An RDF keyword-based query processing tool can be categorized as *schema-based*, when it exploits the RDF schema to compile a keyword-based query into a SPARQL query, or as *graph-based*, when it directly operates on the RDF dataset. We provide a brief review of each of these approaches.

We start with RDF schema-based approaches. Han et al. [9] described an algorithm that assembles a query from the keywords and experiments with DBpedia and Freebase. *QUICK* [17] translates keyword-based queries to SPARQL queries with the help of the users, who choose a set of intermediate queries, which the tool ranks and executes. Gkirtzou et al. [8] described a method to generate candidate SPARQL queries, with natural language descriptions, to help users decide which query to execute.

As for graph-based tools, SPARK [19] uses techniques, such as synonyms from WordNet and string metrics, to map keywords to knowledge base elements. The matched elements in the knowledge base are then connected by minimum spanning trees from which SPARQL queries are generated. The most likely SPARQL query is selected using a probabilistic ranking model that incorporates the quality of the mapping and the structure of the query is proposed. Tran et al. [15] combined the idea of generating summary graphs for the RDF graph, using the class hierarchy, to generate and rank candidate SPARQL queries. Le et al. [12] also proposed to process keyword queries using a summarization algorithm. Zheng et al. [18] also adopted a pattern-based approach. Elbassuoni and Blanco [6] described a technique to retrieve a set of subgraphs that match the keywords, and to rank them based on statistical language models.

3 The Keyword Search Tool

3.1 The Keyword Translation Algorithm

The keyword translation algorithm: (1) accepts a keyword-based query K over an RDF dataset T (or a relational database T), together with its RDF (or relational) schema S; (2) finds matches with the keywords in K; (3) creates an abstract query by exploring the keyword matches found and the schema S; (4) compiles the abstract query into a SPARQL (or SQL query), which is then executed.

Schema Graph. The algorithm uniformly treats the RDF or relational schema S as a labelled schema multigraph $G_S = (N_S, E_S, EL_S)$. In an RDF environment, N_S are the classes, and EL_S labels arcs with object properties or with rdfs:subsetOf in E_S, as in the RDF graph induced by S. In a relational environment, N_S are the relation scheme names, and EL_S labels arcs in E_S with foreign key names, as in the multigraph induced by S.

Computing Keyword Matches. Consider first the RDF environment. The algorithm starts by computing: (1) a set of classes and properties in S whose metadata (i.e., labels and descriptions) match keywords in K; (2) a set of property/value pairs (p, v) such that there is a triple (s, p, v) in T such that v matches a keyword in K.

The algorithm organizes the result of the matches as a set of nucleuses. Each *nucleus* is a triple (*c, PR, VA*), where *c* is a class, *PR* is a possibly empty list of properties, and *VA* is a possibly empty list of property/value pairs, such that: (1) if *PR* and *VA* are empty, *c* must be the result of a metadata match; (2) each property in *PR* has *c* as domain and is the result of a metadata match; (3) each property/value pair (*p, v*) in *VA* is such that the domain of *p* is *c* and (*p, v*) is the result of a data match.

For the relational environment, the algorithm proceeds almost exactly as for the RDF case. It computes: (1) a set of relation scheme and attribute in *S* whose metadata (i.e., names and descriptions) match keywords in *K*; (2) a set of attribute/value pairs whose values match keywords in *K*. A nucleus is again a triple (*c, PR, VA*), where *c* is a relation scheme name, *PR* is a possibly empty list of attributes, and *VA* is a possibly empty list of attribute/value pairs, such that: (1) if *PR* and *VA* are empty, *c* is the result of a metadata match; (2) each element in *PR* is an attribute of *c* and is the result of a metadata match; (3) each attribute/value pair (*p, v*) in *VA* is such that *p* is an attribute of *c* and (*p, v*) reflects a data match.

In either case, RDF or relational, the matching process results in a set of nucleuses.

Synthesizing an Abstract Query. The next stage is to synthesize an abstract query that captures the keyword-based query. It depends on the schema multigraph and the set of nucleuses that the matching process outputs, independently of the environment. To synthesize an abstract query, the translation algorithm implements two heuristics, called the *scoring* and the *minimization* heuristics.

Briefly, the scoring heuristic: (1) considers how good a match is, say the keyword *"south"* matches the literal *"south"* better than the literal *"South Africa"*; (2) assigns a higher score to metadata matches, on the grounds that, if the user specifies a keyword, say *"Desert"*, that matches both a class label (or relation scheme name), say *"Desert"*, and a property (or attribute) value of an instance (or a tuple), say the location *"Sahara Desert"*, then the user is probably more interested in the class (scheme) labelled *"Desert"* than the specific instance *"Sahara Desert"*; (3) assigns a higher score to nucleuses that cover a larger number of keywords. The heuristic is formalized by defining a *score function* for the nucleuses [7].

The minimization heuristic tries to generate minimal answers, in two stages. Ideally, it should try to find the smallest set of nucleuses that covers the largest set of keywords and that has the largest combined score. However, this is an NP-complete problem. The first stage of the minimization heuristic then implements a greedy algorithm that prioritizes the nucleuses with the largest scores that cover a large subset of *K*. The second stage of the minimization heuristic then connects the classes (or relation scheme) in such nucleuses, using a small number of equijoins. This is equivalent to generating a Steiner tree *ST* of the schema graph that covers the classes (or relation scheme) of the prioritized nucleuses. Finally, the algorithm uses the edges of *ST* to generate join clauses, and the nucleuses to generate selection clauses of the abstract query.

Compiling the Abstract Query into a Concrete Query. The final stage of the translation algorithm is to compile the abstract query into a concrete SPARQL or SQL query for the underlying RDF store or relational DBMS. Section 3.2 illustrates this process.

Execution. The tool then executes the concrete query to generate representations of answers to the keyword-based query, which are then passed to the user.

3.2 An Example

This section discusses how the keyword translation algorithm works, using Mondial, whose ER-Diagram is available at https://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-ER.pdf. Figure 1(a) shows the keyword-based query $K = \{Country, Province, Desert, Atacama\}$. Figure 1(b) depicts the structure of an abstract query that corresponds to a Steiner tree of the schema graph of Mondial, where:

- The nodes correspond to classes *Country*, *Province*, *Desert*, and *Geo_Desert*; the first three nodes correspond to nucleuses and the fourth node just completes the tree to link the classes *Province* and *Desert*.
- The arcs represent object properties between such classes.

The classes that correspond to nucleuses reflect the following matches:

- *Country*, *Province*, and *Desert* have an exact metadata match with the labels of classes *Country*, *Province* and *Desert*.
- *Atacama* matches a value of property Label whose domain is the class *Desert*.

Note that the keyword *Desert* is ambiguous since it matches with the labels of classes *Desert* and *Geo_Desert*, but the translation algorithm selects the class *Desert* because the corresponding nucleus has the highest score. Also, the match found with the instance of class *Desert* whose label is *Atacama* does not need to be shown in the abstract query because that information is in the nucleus data.

$K = \{Country, Province, Desert, Atacama\}$

(a) Keyword-based query

(b) Abstract Query

```
1.  SELECT DISTINCT  ?C0 ?C1 ?C2
2.  WHERE{
3.   ?I_C0 rdf:type <http://www.swtech.org/mondial/Country> .
4.   ?I_C1 rdf:type <http://www.swtech.org/mondial/Desert> .
5.   ?I_C2 rdf:type <http://www.swtech.org/mondial/Geo_Desert> .
6.   ?I_C3 rdf:type <http://www.swtech.org/mondial/Province> .
7.   ?I_C2 <http://www.swtech.org/mondial/Geo_Desert#Province> ?I_C3 .
8.   ?I_C3 <http://www.swtech.org/mondial/Province#Code> ?I_C0 .
9.   ?I_C2 <http://www.swtech.org/mondial/Geo_Desert#Desert> ?I_C1
10.  FILTER (orardf:textContains(?C1, "{atacama}", 0) )
11.  ?I_C0 rdfs:label ?C0 .
12.  ?I_C1 rdfs:label ?C1 .
13.  ?I_C3 rdfs:label ?C2  }
```

(c) SPARQL Query

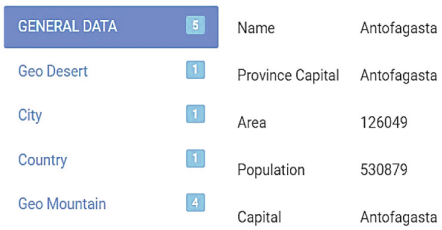
```
1.  SELECT Desert.META_REPCOL,
2.             Desert.NAME,
3.             Country.META_REPCOL,
4.             Country.CODE,
5.             Province.META_REPCOL,
6.             Province.NAME,
7.             Province.COUNTRY
8.    FROM Desert, Country, Province, Geo_Desert
9.  WHERE
10.       ((contains(Desert.META_REPCOL,'{atacama}', 0) > 0)
11.       AND (Geo_Desert.DESERT = Desert.NAME)
12.       AND (Geo_Desert.PROVINCE = Province.NAME)
13.       AND (Province.COUNTRY = Country.CODE)
```

(d) SQL Query

Fig. 1. Sample SPARQL and SQL queries.

	Country	Desert	Province
1	Chile	Atacama	Antofagasta
2	Chile	Atacama	Arica y Parina
3	Chile	Atacama	Atacama
4	Chile	Atacama	Tarapacá

(a) Results for the keyword-based query $K = \{Country, Province, Desert, Atacama\}$.



(b) Navigation panel that results from clicking on the instance *Antofagasta*.

Fig. 2. Sample query results and navigation panel.

From this abstract query, the translation algorithm generates the SPARQL query shown in Fig. 1(c), where:

- Line 1 constructs the result of the query, which consists of the labels of instances of the classes *Country*, *Desert*, and *Province*, respectively, as shown at Fig. 2(a).
- Lines 3 to 6 introduce variables that range over instances of the classes that correspond to the nodes of the Steiner tree.
- Lines 7 to 9 represent relationships between the instances, that correspond to the arcs of the Steiner tree.
- Line 10 represents the match with the keyword *Atacama*.
- Lines 11 to 13 translate instances to their labels, which are user-friendly.

The translation algorithm generates the SQL query shown in Fig. 1(d), where:

- Lines 1 to 7 capture the primary keys and the META_REPCOL columns of the relation schemes. For each relation, the preparation stage adds a column, META_REPCOL, whose values parallel the instance labels of the RDF version.
- Line 8 corresponds to the nodes of the Steiner tree.
- Line 10 represents the match with the keyword *Atacama*.
- Lines 11 to 13 represent equijoins that correspond to the arcs of the Steiner tree.

Note that in the line 10 of both queries, equivalent contains operators of Oracle Text are used to filter the results by the instance of *Desert* with label *Atacama*. In two cases, the contains operators were configured using the default parameters.

Navigation over the Results of a Keyword-Based Query. The tool does not return the results of a keyword-based query as a set of RDF triples or as a set of tuples. After several experiments with the users, we decided to present the results of a keyword-based query in a tabular format for both the RDF and the relational environments, combined with facilities to help users explore the results. For example, Fig. 2(a) shows the result of the keyword-based query in Fig. 1(a). If the user clicks on *Antofagasta*, the tool will display the results shown in Fig. 2(b).

In the RDF environment, navigation over the query results is much simpler than in the relational environment, since it directly crawls the RDF graph. Also, navigation leverages on the ability of SPARQL queries to retrieve data and metadata seamlessly.

To support navigation over data about an instance, the tool uses two queries:

1) SPARQL query Q_1:

```
1.  SELECT ?Class ?Property ?Value ?Label
2.    WHERE {
3.       ?uri  ?prop ?Value .
4.       ?prop  rdfs:label ?Property .
5.       OPTIONAL { ?Value rdfs:label ?Label } .
6.       ?uri  rdf:type ?c .
7.       ?c rdfs:label ?Class   }
```

2) SPARQL query Q_2:

```
1.  SELECT ?Class ?Value ?Label
2.    WHERE {
3.       ?Value ?property ?uri .
4.       OPTIONAL {?Value rdfs:label ?Label } .
5.       ?Value rdf:type ?c .
6.       ?c rdfs:label ?Class
7.  }
```

We assume that all classes, instances of classes, and properties have a mandatory label. Consider query Q_1, and assume that variable ?uri in Line 3 binds to instance *I*. The query retrieves any property and its value that *I* has (Line 3), the label of such property (Line 4), the label of the value, if it corresponds to an object property value (Line 5), any class that I belongs to (Line 6), and the label of such class (Line 7). Query Q_2 retrieves an instance *J* that is related to *I* by an object property (Line 3), the label of *J*, if it exists (Line 4), a class that *J* belongs to (Line 5), and the label of such class (Line 6). The results of both queries are then post-processed to generate a navigation panel. Figure 2(b) illustrates the navigation panel that results from clicking on *Antofagasta*.

However, it is not as simple to explore the query results in the relational environment as is in the RDF environment. The tool implements the following strategy:

(1) Construct a SQL query, using the template:

SELECT pk_1, ..., pk_n, Label, p_1, p_2, ..., p_n FROM *T* WHERE <*instance filter*>

that queries table *T* to retrieve the primary keys, instance label, and properties, and whose WHERE clause is a filter defined by the identifier created for the target instance. In our example, the filter is built using the primary key values for instance *Antofagasta*.

(2) Compute a set *JF* of tables such that $V \in JF$ iff *T* has a foreign key to *V*. For each $V \in JF$, a SQL query with an inner join between *T* and *V* and a TARGET clause composed of the primary keys and the label columns of *V* is compiled and executed.

(3) Compute a set *JT* of tables such that $U \in JT$ iff *U* has a foreign key to *T*. For each table $U \in JT$, a SQL query with an inner join between *T* and *U* and a TARGET clause composed of the primary keys and the label columns of *U* is compiled and executed.

(4) Obtain the labels of the properties retrieved in (a), the label of *T*, and the labels of the tables in $JF \cup JT$.

The data in Fig. 2(b) is built by post-processing the results of steps (3) and (4).

In our running example, the generation of the panel in Fig. 2(b) required 2 SPARQL queries over the RDF graph, which ran in 0.89 s, while it required 9 SQL queries over the relational database, which ran in 2.75 s. Thus, navigation over the RDF graph was nearly 3 times faster than in the relational database.

4 Experiments

Experimental Setup. *QUIOW* was implemented as a RESTful Web application developed in Java, and ran on Windows 7 Ultimate, using a quad-core processor Intel (R) Core(TM) i5-2450 M CPU @ 2.50 GHz, 4 GB of RAM. The relational databases and RDF datasets were stored in Oracle 12c, running on a quad-core processor Intel(R) Core(TM) i5 CPU 660 @ 3.33 GHz, 7 GB of RAM, and 4,096 KB of cache size. Table 1 shows basic statistics about the RDF datasets and relational databases used in the experiments. We ran the Coffman's benchmark [5] using the relational databases and their triplified versions of the Mondial dataset (available at https://www.dbis.informatik.uni-goettingen.de/Mondial/) and the full and more recent relational version of IMDb (available at https://sites.google.com/site/ontopiswc13/home/imdb-mo), which we refer to as Full IMDb to differ it from the Restricted IMDb version used in [5].

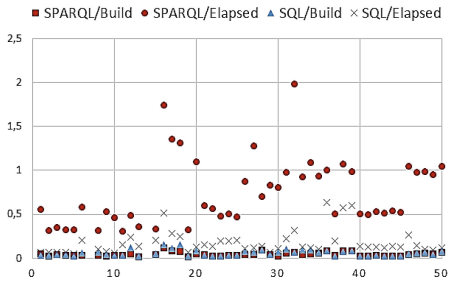
We measured the query build time – the time taken to process matches and construct the SQL or SPARQL query, and total elapsed time – the time from the submission of the keyword-based query until the display of the first 75 results.

We used a separated tool to index the relational databases to support keyword search and to triplify the relational databases via a set of relational-to-RDF mappings and to create indexes over the RDF dataset to support keyword search. In this case, we measured the time to index the relational database and to create the RDF dataset.

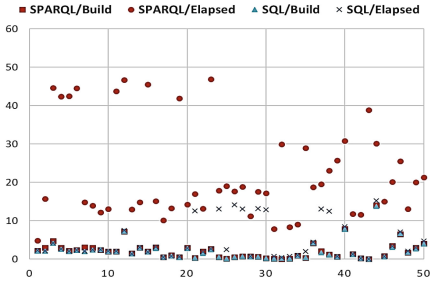
Table 1. Statistics of Mondial and IMDb datasets.

RDF Dataset	#Triples	
	Mondial	IMDb
Classes	40	11
Object properties	62	11
Datatype properties	130	25
Indexed properties	71	7
Indexed prop. instances	11,094	12,609,418
Class instances	43,869	70,520,744
Object prop. instances	63,652	204,917,673
Total number of triples	**235,387**	**382,295,213**

Relational Database	#Objects	
	Mondial	IMDb
# of Relations	40	11
# of Attributes	187	20
# of Tuples	40,247	70,520,722
Size (in GB)	**0.11**	**60.63**



(a) Experiments with Mondial



(b) Experiments with IMDb

Fig. 3. Build time and total elapsed time.

Experiments with the Coffman's Benchmark Datasets. Figure 3 shows the execution times of the keyword-based queries defined in Coffman's benchmark. The Y-axis represents the query build time and the total elapsed time, in seconds, of each query in the benchmark, numbered 1 to 50 on the X-axis. Note that, for each keyword-based query: the SPARQL total elapsed time (shown as a dot) was always larger than the SQL total elapsed time (shown as a cross), and the SPARQL and the SQL query build times (respectively shown as squares and triangles) were nearly the same (most squares are on top of the triangles).

To reduce ambiguity when using the Full IMDb, and consequently to improve processing time, we surrounded most keywords with quotes. For instance, consider the query {*denzel washington*}. If we treat the keywords separately, we find that {*denzel*} has 670 data matches, while {*washington*} has 23,720. Indeed, "*washington*" is a very ambiguous keyword, since it matches the name of an actor, city, state, etc. Hence, if we treat the query as "*denzel* OR *washington*" we have a total of 23,851 data matches. However, if we treat the query as "*denzel* AND *washington*", we have only 539 data matches. As pointed out in [7], this affects the computation of the scores. Indeed, using quotes, the total elapsed times of the SQL query to compute the value score for class *movie_info* and property *info* was 0.14 s. By contrast, without quotes, the total elapsed time was 5.04 s. Since we execute queries for all distinct domains and properties that match at least one of the keywords, the total time to create an abstract query in this example was, on average, 30 times faster with quotes. We note that the use of quotes only improved the query build time and did not change the final results.

Table 2 shows the average (Avg), maximum (Max) and minimum (Min) of the total elapsed time and the query build time of the 50 queries in Coffman's benchmark, for the relational and RDF variations of Mondial and IMDb.

Table 2. Summary of the experiments with Mondial and IMDb datasets.

		Mondial	IMDb	Mondial	IMDb	Mondial	IMDb
		Total elapsed time (in sec)		Query build time (in sec)		Query build time/Total elapsed time	
Avg.	Relational	0.147	4.360	0.066	2.128	50.5%	71.2%
	RDF	0.802	22.273	0.047	2.263	6.6%	11.2%
Max.	Relational	0.516	15.279	0.154	13.765	73.6%	98.8%
	RDF	1.982	46.868	0.121	14.016	13.5%	46.4%
Min.	Relational	0.051	0.093	0.021	0.064	19.6%	1.9%
	RDF	0.311	7.895	0.012	0.030	3.3%	0.1%

Preparation and Maintenance. In the relational environment, preparing Mondial to support keyword search took less than a minute, while preparing IMDb took 60 min. Thus, re-indexing would be a feasible strategy to maintain the relational version of Mondial, but slow for IMDb. Triplifying Mondial, which is a small database, took 5 min; thus, full retriplification would be a feasible strategy to maintain the RDF version. However, triplifying IMDb was very slow, taking 5 h. This calls for an incremental strategy to maintain complex RDF datasets, such as IMDb.

Quality of the Query Results. For each keyword search executed, the results obtained were exactly the same in both the RDF and relational environments, as expected, since the construction process of the abstract query was the same in both cases. In this aspect, the difference is in the concrete query structure (SPARQL versus SQL) and not in the query target. The correctness of the results of the translation process was satisfactory, for Mondial and IMDb, in both environments, as compared with Coffman's benchmark. For Mondial, the tool correctly answered 33 queries, nearly 65%. For IMDb, 36 of 50 queries were correctly answered, approximately 72%.

Query Build Time. In all experiments, the query build time was nearly the same in both environments, since processing matches and constructing the abstract query were basically the same in both cases. In the relational environment, for the experiments with Mondial, the query build time accounted for 40–50% of the total elapsed time, on average; for the experiments with IMDb, it raised to slightly over 70–75%, possibly due to the ambiguity of IMDb data. By contrast, in the RDF environment, the query build time accounted for only 6–15% of the total elapsed time, on average. This behavior can be explained because matching is a costly process in both environments, but SPARQL queries take much longer to execute than SQL queries.

Total Elapsed Time. The total elapsed time was reasonable, on average, in all experiments. Even for a large database, such as IMDb, the total elapsed time was, on average, nearly 4 s, in the relational environment, but raised to 22 s, in the RDF environment. Indeed, the total elapsed time of the SQL queries was 4–6 times faster than the SPARQL queries, on average. Queries with contains filter use a text index, which is over all object values of the triples, for RDF datasets. But, for relational databases, there is a separate, smaller index for each text attribute. Thus, the elapsed time of SQL queries with a contains filter was smaller than that of SPARQL queries.

Navigation. We are not aware of any benchmark to evaluate this aspect, which is closely related to the users' interests. From the experiments (not detailed here), we may conclude that navigation through the results was much slower in the relational environment, since it involved several joins, as discussed in Sect. 3.2.

Navigation versus Querying. The previous observations suggest that the RDF environment should be favored when users frequently navigate over the keyword-based query results. Being reasonable in all experiments, the total elapsed time should not be an a priori argument to avoid the RDF environment.

5 Conclusions and Future Work

We described *QUIOW*, a tool designed to support keyword-based query processing for both RDF datasets with schemas and relational databases. Using full version of IMDb and the Mondial databases, the experiments indicated that the total elapsed time was quite reasonable, on average, in both environments. Also, the experiments permitted us to conclude that the relational version reached better query performance, but had a poor navigation performance, when compared with the RDF version. Thus, if users tend to first query the data and then navigate through the results, the RDF version is an

interesting alternative. The experiments suggest, as future work, to improve the performance of the keyword matching process by using alternative technologies, or by parallelization. We also plan to expand the tool to support other RDF stores and relational systems. Finally, we plan to explore users' preferences to deal with databases with very large schemas and use a domain ontology to expand keywords.

References

1. Aditya, B., et al.: BANKS: browsing and keyword searching in relational databases. In: VLDB 2002, pp. 1083–1086 (2002)
2. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: a system for keyword-based search over relational databases. In: ICDE 2002, pp. 5–16 (2002)
3. Bast, H., Buchhold, B., Haussmann, E.: Semantic search on text and knowledge bases. Found. Trends® Inf. Retrieval **10**(2–3), 119–271 (2016)
4. Bergamaschi, S., et al.: Combining user and database perspective for solving keyword queries over relational databases. Inf. Syst. **55**, 1–19 (2016)
5. Coffman, J., Weaver, A.C.: A framework for evaluating database keyword search strategies. In: CIKM 2010, pp. 729–738 (2010)
6. Elbassuoni, S., Blanco, R.: Keyword search over RDF graphs. In: CIKM 2011, pp. 237–242 (2011)
7. García, G.M., Izquierdo, Y.T., Menendez, E., Dartayre, F., Casanova, M.A.: RDF keyword-based query technology meets a real-world dataset. In: EDBT 2017, pp. 656–667 (2017)
8. Gkirtzou, K., Papastefanatos, G., Dalamagas, T.: RDF keyword search based on keywords-to-SPARQL translation. In: NWSearch 2015, pp. 3–5 (2015)
9. Han, S., Zou, L., Xu Yu, J., Zhao, D.: Keyword search on RDF graphs - a query graph assembly approach. arXiv:1704.00205 [cs.DB] (2017)
10. He, H., et al.: BLINKS: ranked keyword searches on graphs. In: SIGMOD 2007, pp. 305–316 (2007)
11. Hristidis, V., Papakonstantinou, Y.: DISCOVER: keyword search in relational databases. In: VLDB 2002, pp. 670–681 (2002)
12. Le, W., Li, F., Kementsietsidis, A., Duan, S.: Scalable keyword search on large RDF data. IEEE TKDE **26**(11), 2774–2788 (2014)
13. Li, G., Ji, S., Li, C., Feng, J.: Efficient type-ahead search on relational data: a tastier approach. In: SIGMOD 2009, pp. 695–706 (2009)
14. Oliveira, P., Silva, A., Moura, E.: Ranking candidate networks of relations to improve keyword search over relational databases. In: ICDE 2015, pp. 399–410 (2015)
15. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: ICDE 2009, pp. 405–416 (2009)
16. Yu, J., Qin, L., Chang, L.: Keyword Search in Databases. Morgan and Claypool, San Rafael (2009)
17. Zenz, G., Zhou, X., Minack, E., Siberski, W., Nejdl, W.: From keywords to semantic queries - incremental query construction on the semantic web. Web Semant.: Sci. Serv. Agents World Wide Web **7**(3), 166–176 (2009)
18. Zheng, W., Zou, L., Peng, W., Yan, X., Song, S., Zhao, D.: Semantic SPARQL similarity search over RDF knowledge graphs. Proc. VLDB Endow. **9**(11), 840–851 (2016)
19. Zhou, Q., Wang, C., Xiong, M., Wang, H., Yu, Y.: SPARK: adapting keyword query to semantic search. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 694–707. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_50

An Abstract Machine for Push Bottom-Up Evaluation of Datalog

Stefan Brass(✉) and Mario Wenzel

Institut für Informatik, Martin-Luther-Universität Halle-Wittenberg,
Von-Seckendorff-Platz 1, 06099 Halle (Saale), Germany
{brass,mario.wenzel}@informatik.uni-halle.de

Abstract. The Push Method for Bottom-Up Evaluation in deductive databases was previously defined as a translation from Datalog to C++. Performance tests on some benchmarks from the OpenRuleBench collection gave very encouraging results. However, most of the systems used for comparison compile the query into code of an abstract machine and then use an emulator for this code. Therefore, runtimes cannot be directly compared. In this paper, we propose an abstract machine for bottom-up evaluation of Datalog based on the Push Method. This also helps to clarify some optimizations we previously expected from the C++ compiler. First tests have shown that the code running in the emulator of the abstract machine is only 1.5 times slower than the native code generated by the C++ compiler. This is better compared to the case for Prolog.

1 Introduction

The database language SQL is a very successful declarative language, but usually only parts of applications are developed in SQL, the rest is written in a standard language like Java or even PHP. The purpose of deductive databases is to increase the declaratively specified part of an application (ideally to 100% for many applications). This will lead to greater productivity, fewer bugs, and more independence from specific hardware architectures.

Deductive databases use some variant of Datalog, based on a pure and simple subset of the logic programming language Prolog. Datalog is more than SQL plus recursive views, because it permits programming. A reason for the current revival of Datalog is that it is now used also for applications that are not traditional database applications, such as static analysis of program code [12], cloud computing and semantic web applications. The commercial deductive database system LogicBlox [1] is successful probably because it offers many functions in an integrated system with only one language. In the previous DEXA conference, we proposed Datalog extensions that can be used for writing a Datalog compiler in Datalog [5]. This also continues our previous work on declarative output [4].

Over the years, a lot of work has been done on recursive query evaluation in deductive databases. In particular, the magic set method is a well known source-to-source optimization to make bottom-up evaluation goal-directed. However,

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 270–280, 2018.
https://doi.org/10.1007/978-3-319-98812-2_23

after the transformation, one still needs a pure bottom-up engine that computes all facts that are derivable from the given database facts and the program rules. Theoretically, this is simple. Yet, many deductive database prototypes have failed to deliver a competitive performance. For instance, XSB, a Prolog system with tabling, clearly beats the CORAL deductive database [11].

In [3], we did performance comparisons for different implementations of bottom-up evaluation in main memory. One algorithm, the "Push Method", was further developed in [6,7]. It applies the rules from body to head as any form of bottom-up evaluation, but it immediately "pushes" a derived fact to other rules with matching body literals. In this way, derived facts often do not have to be materialized, and temporary storage can be saved. It can be seen as an extreme form of seminaive evaluation that dates back to the PhD thesis of Schütz [13]. "Pushing" tuples has also become an attractive technique for standard databases [10]. It seems well suited for modern hardware because it keeps the actively used set of data small.

The Push Method was defined as a translation from Datalog to C++. Performance tests on some benchmarks from the OpenRuleBench collection [9] gave encouraging results. However, most of the systems used for comparison compile the query into code of an abstract/virtual machine which is then interpreted. Since we compiled to native code, runtimes were not directly comparable. Experiences with compiling Prolog to machine code [8] suggest that this gives approximately a factor of 3 (the range in that paper was between 1.3 and 5.6). In a first test, our abstract machine implementation was only 1.5 times slower than the native code approach. This strengthens our previous performance claims.

For Prolog systems, there is usually first an implementation with an abstract machine. Later, some systems add a compilation to native machine code in order to increase the performance. In our case, we had a method for translating to native code, and now propose an abstract machine. Besides making the performance more comparable in benchmarks, there are other advantages of this:

- The user does not need to install a C++ compiler, and problems due to different compiler versions are avoided.
- Programs can be more easily distributed for multiple platforms.
- The compilation time is reduced (it is faster to compile to code of the abstract machine, than to compile first Datalog to C++, and then C++ to machine code). Fast compilation is especially useful during development.
- Compiling to code of an abstract machine gives better control over optimizations. In [6] we did optimizations such as partial evaluation in the generation of C++ code. In [7], we used a much simpler translation, because the C++ compiler actually does many optimizations. However, this is hidden in the C++ compiler, and depends on the compiler version and chosen options. With the abstract machine, our optimizations become explicit again.
- The abstract machine may also be a first step towards direct native code generation via the LLVM library. In [10], it is noted that this gives better performance than code generation via C++.

– If a user trusts the emulator for the abstract machine, he/she does not need to trust the code for a specific application, whereas binary code is inherently more dangerous.

2 Example Application: Transitive Closure

The input language is basic Datalog, i.e. pure Prolog (Horn clauses) with only constants and variables, but no function symbols. Obviously, the language should be extended later. But for first performance tests to check the approach, this basic language is sufficient. As an example, consider the well-known transitive closure program, which is one benchmark from the OpenRuleBench collection [9]:

```
tc(X, Y) :- par(X, Y).
tc(X, Z) :- par(X, Y), tc(Y, Z).
```

The query is `tc(X,Y)`, i.e. all derivable facts should be computed. Other benchmarks check goal-directed computation, and we have also good results for the query `tc(1,X)` by applying our SLDMagic transformation [2], but the focus in this paper is on pure bottom-up evaluation.

The predicate `par` is a database predicate. The benchmark contains files with facts for this "parent" relation, such as `par(1,2)`. The smallest file contains 50 000 facts. Database predicates are considered the input of the program. They are loaded into main memory relations at the beginning of program execution. We require that they are declared with argument types:

```
db par(int, int) facts 'par.dl'.
```

The system contains a collection of "table data structures". A table data structure permits to store a set of tuples (a relation) and to iterate over a subset of its tuples, given values for some columns. The input and output columns that a specific table data structure supports are described by a "binding pattern", i.e. a string of characters "`b`" ("bound") or "`f`" ("free"), where the i-th character corresponds to the i-th column. "Bound" columns have known values when the table is accessed. We use such table data structures matching the required binding patterns in the rules. In the example, the loader will store the `par` facts in two table data structures:

– In a list `par_ff` that permits to iterate over all `par` facts (for the first rule).
– In a multi-map (e.g. a hash table) `par_fb` for the literal `par(X,Y)` in the second rule. This data structure permits to iterate over all `X`-values given a value for `Y` (binding pattern "free, bound"). The Push Method will activate the second rule when a new fact for the body literal `tc(Y,Z)` is found, therefore we know a value for `Y` when we access `par` here.

One can use a table data structure with a more general binding pattern (a subset of "`b`") for a given body literal, e.g. do a full table scan with `par_ff` instead of an index access with `par_fb`. Then one must filter out non-matching tuples.

We assume that the user specifies for which predicates duplicate elimination should be done. This is basically the same as the "`table`" declaration in XSB and similar logic programming systems. In order to guarantee termination, every recursive cycle in the predicate dependency graph must contain at least one predicate for which duplicate elimination is activated. In the example, the `par`-relation might be cyclic, therefore we require duplicate elimination for `tc`. This means that a set data structure `tc_bb` will be used.

3 The Push Method

The idea of the Push Method is that the producer of derived facts has the control and actively "pushes" these facts to the consumer (rules in which the produced facts match body literals). In [3], we contrasted it with the standard "Pull" method where the consumer fetches the next fact when needed.

One creates a procedure for each derived predicate p that is called whenever a fact $p(c_1, \ldots, c_n)$ is derived. This procedure has to ensure that all rule instances with $p(c_1, \ldots, c_n)$ in the body are eventually applied. This is simple for rules that contain only a single body literal with a derived predicate ("linear rules"). Since the complete relations for the other (database) body literals are known, the necessary joins, selections and projections can be immediately done in order to perform the procedure calls corresponding to the head of the rule. For instance, the procedure for the "transitive closure" program looks as follows:

```
void tc(int c1, int c2) {
    // Is this fact a duplicate?
    if(!tc_bb.insert(c1, c2)) // set insert fails if member
        return;

    // This is a query predicate, store answer:
    tc_ff.insert(c1, c2); // List data structure

    // Rule tc(X, Z) :- par(X, Y), tc(Y, Z):
    int Y = c1;
    int Z = c2;
    foreach X in par_fb(Y) do
        tc(X, Z);
}
```

The procedure for a predicate p contains one code block for each rule that contains the predicate p in the body (in this case, it is only one rule).

For "complex rules" that have more than one body literal with a derived predicate, temporary storage of derived facts seems unavoidable. While for special cases, optimizations are possible, in general one creates a temporary table for each body literal with a derived fact. When a rule is activated for a new fact for a specific body literal, one applies all rule instances with this fact and

the previously derived facts stored in the temporary tables for the other body literals. As explained in [13], this can be seen as an extreme form of seminaive evaluation, where the "delta" consists of a single fact.

Finally, a procedure "`start`" executes all rules without derived predicates in the body. It is called by the main program after loading the database predicates:

```
void start() {
    // Rule tc(X, Y) :- par(X, Y):
    foreach (X, Y) in par_ff do
        tc(X, Y);
}
```

4 The Bottom-Up Abstract Machine

4.1 Design Principles

Argument Values Are Passed in Registers, Not on the Stack: The Push Method as explained above does a lot of procedure calls (one for each derived fact including possible duplicates). Therefore, procedure calls should be fast. Argument values are passed in registers of the abstract machine, not on the stack. Only values that are overwritten in recursions must be saved on the stack and later restored. For instance, consider again the rule

`tc(X, Z) :- par(X, Y), tc(Y, Z).`

This rule is executed when a fact was derived matching the underlined body literal. Suppose that the first argument (`Y`) is stored in Register 0, the second (`Z`) in Register 1. By looping over `par`-facts, we have to create calls for the rule head. But the second argument of the head literal is also `Z`, so we do not have to touch Register 1. Only Register 0 must be loaded with a new value (`X`) before the call. Therefore its previous contents is saved on a stack, and later restored.

It is also not required that the first argument is stored in Register 0. There can be several specializations of the same predicate procedure depending on where the arguments are stored or whether the arguments are known constants.

Strings Are Mapped to Integers: There are string tables that implement bijective mappings from strings appearing in the data to integers. For distinct domains, different tables are used in order to get small, sequential numbers.

The Interpretation Overhead Should Be Small: The emulator of the abstract machine is in effect an interpreter for the machine code. The overhead for interpretation should be small. This implies that the granularity of the machine instructions should be large. The real work that the interpreter does, once a "machine instruction" has been decoded, is often basically the same C++ code that was contained in the compiled version.

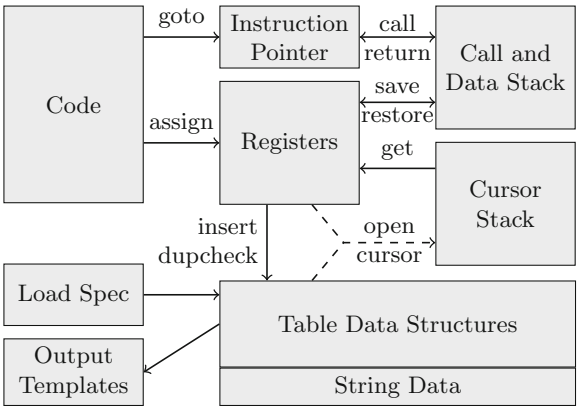


Fig. 1. Memory architecture of the push bottom-up abstract machine

For instance, the duplicate check at the beginning of the `tc` procedure (Sect. 3) is a single machine instruction. If we would use single instructions for the insertion, the `if`, and the `return`, we would only increase the interpretation overhead and make the code larger. In the same way, a entire control of a loop over a set of tuples is done in a single instruction (plus one for initialization).

Compact Instructions: Instructions are small to make the code more cache-friendly. This implies that instructions have variable length (an opcode and arguments as needed). There are also special instructions for small operands.

Special Instructions for Common Cases: For accessing tables with only a few columns, there are special instructions. This permits to compile more constants in the code and unroll loops: Instead of a loop over the columns, where the body is executed two times, two statements are generated.

4.2 Memory Areas of the Bottom-Up Abstract Machine

Data values, i.e. values of variables in the rules and predicate arguments, are stored in registers. Each register can contain a single integer. The number of necessary registers is determined when a Datalog program is compiled.

There is a joint stack for return addresses of procedure calls and for storing saved register values. A second stack contains cursors, i.e. data structures used for iterating over a subset of the tuples in a table data structure. Since we use only nested loop and index joins, only the topmost cursor in the stack is accessed.

Before program execution starts, the load specification is used to create and fill the table data structures with data from the input data files. It also determines the binding pattern(s) for each table. When data is loaded, also a selection and projection can be done for each table. Then the program computes derived

Basic Instructions	
`NULL`	Does nothing. Can be used to fill space (for alignment).
`HALT`	Finishes program execution.
Procedure Calls	
`CALL` a	Push instruction pointer on stack, jump to address a.
`RETURN`	Pop code address from stack, jump to that address.
Inserting Tuples Into Relations, Duplicate Check	
`INSERT_`N t, v	Insert tuple in registers v into table t
`DUPCHECK_`N t, v	Insert tuple in registers v into table t, return if already there.
Saving, Restoring, and Copying Register Values	
`SAVE_REG` r	Push value of register r on stack.
`RESTORE_REG` r	Pop value of register r from stack.
`COPY` r_1, r_2	Copy the value of r_2 into r_1
`ASSIGN` r, i	Store value i into register r
Loops over Cursors, Accessing Values from Cursors	
`LOOP_`B t, v, a	Open cursor on table t with parameters v (values of bound cols). Get first tuple, put cursor on stack. If none, jump to a.
`END_LOOP_`B a	Move cursor on top of stack to next row. If successful, jump to a (start of loop body). Else close and pop.
`GET_`B`_`C r	Store value of column C of current row in top cursor into reg. r.
Conditions, Jumps	
`IF_EQ_RI` r, i, a	If the value of register r is not i, jump to address a.
`IF_EQ_RR` r_1, r_2, a	If the values of r_1 and r_2 are different, jump to a.
`IF_ELEM_`N t, v, a	If tuple in registers v is not contained in table t, jump to a.
`GOTO` a	Unconditional jump to address a.

Fig. 2. Instructions of the push bottom-up abstract machine

facts for output predicates and stores them in table data structures. The output is then generated from this information by means of output templates [4,5].

4.3 Instructions of the Bottom-Up Abstract Machine

The instructions of the bottom-up abstract machine are listed in Fig. 2. There is no space to discuss each instruction in detail, but we will look at the code for the running example and explain how it works. The argument types are:

- a: Code address.
- r: Register number.
- i: Constant data value (integer).
- t: ID (number) of a table data structure.
- C: Column number.
- N: Number of columns of a table.
- B: Binding pattern of a cursor to iterate over a set of tuples in a table.

– v: Register vector, consisting of n register numbers (where n is either N in the same instruction, or number of bound positions in B, or explicitly given).

If N, B and C are small, they are encoded in the opcode (to be compiled into the emulator code). For larger values, the length of the register vector is an extra parameter, and B is selected by an index into a list of binding patterns supported by the table data structure. N and B may also select an implementation variant.

4.4 Machine Program for the Transitive Closure Example

The machine code for the transitive closure example is shown in Fig. 3. The program consists of 15 instructions, and can be executed in our first (experimental) prototype. The example uses the following table data structures:

Table data structures

ID	Table	Comment
0	`par_ff`	Use of `par` in first rule
1	`par_fb`	Use of `par` in second rule
2	`tc_bb`	For duplicate check
3	`tc_ff`	Result

The first part, from address `0` to `14`, corresponds to the procedure `start` (see Sect. 3). It contains the loop over the `par`-facts (stored in the `par_ff` list). The parameters of the instruction at the head of the loop are the ID of the table data structure and the code address just after the loop. The instruction opens the cursor on the stack and fetches the first tuple. If this fails (i.e. the list is empty), it closes the cursor, removes it from the stack and jumps over the rest of the loop. The instruction at the end of the loop fetches the next tuple from the cursor on top of the cursor stack. If that is successful, it repeats the loop by jumping to the beginning to the loop body. Otherwise it closes the cursor, pops it from the stack, and continues with the next instruction after the loop.

The `GET`-instructions take the current values of column 1 and 2 from the cursor on top of the stack, and store them in Register 0 and 1. Thus, these registers contain the arguments of a derived `tc`-tuple, and we call procedure `tc`.

This starts at address 15 with the duplicate check instruction (see Subsection 4.1). It has parameters for the set data structure (`tc_bb`, ID `2`) and the two registers. If the tuple is known already, the procedure immediately returns. Otherwise, the derived tuple is inserted into the result table `tc_ff` (ID `3`). Then Register `0` is saved on the stack, since it will be overwritten. It might be helpful to write the recursive rule as

```
tc(X_new, Y) :- par(X_new, X), tc(X, Y).
```

When the procedure is called, `X` is in Register `0` and `Y` in Register `1`. Therefore, a cursor over the multimap data structure `par_fb` is opened, with Register `1` (`Y`) as a parameter. Getting the current value of the cursor gives the value for `X_new`, which is stored in Register `0`. Then the procedure calls itself recursively.

```
// Procedure start: tc(X, Y) :- par(X, Y).
 0: LOOP_FF 0, 14      // Loop over par_ff (ID 0), if empty goto 14
 4: GET_FF_1 0         // Reg[0] = X from par_ff cursor (Column 1)
 6: GET_FF_2 1         // Reg[1] = Y from par_ff cursor (Column 2)
 8: CALL 15            // Call tc(Reg[0],Reg[1]) (Start address 15)
11: END_LOOP_FF 4      // If next par-tuple exists goto 4
14: HALT               // End of "main" procedure start

// Procedure tc(Reg[0],Reg[1]): tc(X, Z) :- par(X, Y), tc(Y, Z).
15: DUPCHECK_2 2, 0, 1 // If (Reg[0],Reg[1]) in tc_bb (ID 2): return
19: INSERT_2 3, 0, 1   // Store result tuple in tc_ff (ID 3)
23: SAVE_REG 0         // Reg[0] will be overwritten, save it on stack
25: LOOP_FB 1, 0, 38   // Loop over par(X,Y) given Y=Reg[0] (par_fb: ID 1)
30: GET_FB_1 0         // Store X with par(X,Y) (Column 1) in Reg[0]
32: CALL 15            // Recursive call: tc(Reg[0],Reg[1])
35: END_LOOP_FB 30     // If next par(X,Y) tuple exists: goto 30
38: RESTORE_REG 0      // Restore register Reg[0] that was changed in loop
40: RETURN             // End of procedure tc
```

Fig. 3. Machine code for `tc` example

5 Performance

In our previous performance comparisons of the Push Method with benchmarks from the OpenRuleBench suite [9], we have assumed that a factor of 3 must be attributed to the compilation to machine code. The results were still encouraging. Now the important question was whether an interpreted version of abstract machine code is not worse. Fortunately, the first benchmark we were able to execute with our experimental prototype, namely the transitive closure example, is only 1.5 times slower than the native code version:

System	Load	Execution	Total time	Factor	Memory
Push (Switch)	0.004 s	1.145 s	1.147 s	1.0	23.535 MB
Push (Proc.)	0.004 s	1.176 s	1.177 s	1.0	31.392 MB
Push (Abstr.M.)	0.004 s	1.714 s	1.713 s	1.5	31.397 MB
Seminaïve	0.004 s	2.225 s	2.227 s	1.9	31.360 MB
XSB	0.239 s	4.668 s	5.103 s	4.4	135.693 MB
YAP	0.240 s	10.432 s	10.840 s	9.5	147.544 MB
DLV	(0.373 s)	—	51.660 s	45.0	513.748 MB
Soufflé (SQlite)	(0.113 s)	—	11.240 s	9.8	43.083 MB
(compiled)	(0.030 s)	—	0.797 s	0.7	3.867 MB

Transitive Closure Benchmark `tc`(_, _), 50 000 `par`-facts (cyclic) [9]

Another example, the Join1 benchmark of [9], gives the same factor. Here we are able to beat single core compiled Soufflé with our compiled version, and are only minimally worse with our interpreted version. Probably the data structure decides: Soufflé uses a trie, we used extensible hash tables for the `tc` benchmark, but bitmaps for the duplicate check the Join1 benchmark.

6 Conclusion

Both, the translation of Datalog to C++ (and possibly other languages in future), and the translation to code of an abstract machine, have advantages of their own.

For instance, a Datalog system with an abstract machine can work standalone, and does not need a C++ compiler. Lower level optimizations can be studied better with the abstract machine than with the generation of readable C++ code and relying on optimizations of the compiler for that language. Distribution of applications as abstract machine code is simpler.

Vice versa, generated C++ code for the data management can be more easily combined with handwritten code for other parts of the application. And the resulting native code is faster.

However, native code is only slightly faster. The tests reported in this paper showed that the runtime of the benchmarks in the proposed abstract machine were only 1.5 times slower than native code. The current state of the project is reported at: [http://www.informatik.uni-halle.de/~brass/push].

References

1. Aref, M., et al.: Design and implementation of the LogicBlox system. In: Proceedings of the SIGMOD 2015, pp. 1371–1382. ACM (2015). https://developer.logicblox.com/wp-content/uploads/2016/01/logicblox-sigmod15.pdf
2. Brass, S.: SLDMagic — the real magic (with applications to web queries). In: Lloyd, J., et al. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 1063–1077. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44957-4_71
3. Brass, S.: Implementation alternatives for bottom-up evaluation. In: Hermenegildo, M., Schaub, T. (eds.) Technical Communications of the 26th International Conference on Logic Programming (ICLP 2010). LIPIcs, vol. 7, pp. 44–53. Schloss Dagstuhl (2010). http://drops.dagstuhl.de/opus/volltexte/2010/2582
4. Brass, S.: Order in Datalog with applications to declarative output. In: Barceló, P., Pichler, R. (eds.) Datalog 2.0 2012. LNCS, vol. 7494, pp. 56–67. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32925-8_7
5. Brass, S.: Language constructs for a Datalog compiler. In: Benslimane, D., Damiani, E., Grosky, W.I., Hameurlain, A., Sheth, A., Wagner, R.R. (eds.) DEXA 2017. LNCS, vol. 10438, pp. 130–140. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64468-4_10
6. Brass, S., Stephan, H.: Bottom-up evaluation of Datalog: preliminary report. In: Schwarz, S., Voigtländer, J. (eds.) Proceedings of the WLP 2015/2016/WFLP 2016. EPTCS, vol. 234, pp. 13–26. Open Publishing Association (2017). https://arxiv.org/abs/1701.00623
7. Brass, S., Stephan, H.: Pipelined bottom-up evaluation of Datalog programs: the Push method. In: Petrenko, A.K., Voronkov, A. (eds.) PSI 2017. LNCS, vol. 10742, pp. 43–58. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74313-4_4. http://www.informatik.uni-halle.de/ brass/push/publ/psi17.pdf
8. Costa, V.S.: Optimising bytecode emulation for Prolog. In: Nadathur, G. (ed.) PPDP 1999. LNCS, vol. 1702, pp. 261–277. Springer, Heidelberg (1999). https://doi.org/10.1007/10704567_16

9. Liang, S., Fodor, P., Wan, H., Kifer, M.: OpenRuleBench: an analysis of the performance of rule engines. In: Proceedings of the 18th International Conference on World Wide Web (WWW 2009), pp. 601–610. ACM (2009). http://rulebench.projects.semwebcentral.org/
10. Neumann, T.: Efficiently compiling efficient query plans for modern hardware. Proc. VLDB Endow. **4**(9), 539–550 (2011). http://www.vldb.org/pvldb/vol4/p539-neumann.pdf
11. Sagonas, K., Swift, T., Warren, D.S.: XSB as an efficient deductive database engine. In: Snodgrass, R.T., Winslett, M. (eds.) Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD 1994), pp. 442–453 (1994). http://user.it.uu.se/~kostis/Papers/xsbddb.html
12. Scholz, B., Jordan, H., Subotić, P., Westmann, T.: On fast large-scale program analysis in Datalog. In: Proceedings of the 25th International Conference on Compiler Construction (CC 2016), pp. 196–206. ACM (2016)
13. Schütz, H.: Tupelweise Bottom-up-Auswertung von Logikprogrammen (Tuple-wise bottom-up evaluation of logic programs). Ph.D. thesis, TU München (1993)

Novel Database Solutions

What Lies Beyond Structured Data? A Comparison Study for Metric Data Storage

Pedro H. B. Siqueira[1(✉)], Paulo H. Oliveira[2], Marcos V. N. Bedo[3], and Daniel S. Kaster[1]

[1] Department of Computer Science, University of Londrina (UEL), Londrina, Brazil
pedro.braga.siqueira@gmail.com, dskaster@uel.br
[2] Institute of Mathematics and Computer Sciences, University of São Paulo (USP), São Carlos, Brazil
pholiveira@usp.br
[3] Fluminense Northwest Institute, Fluminense Federal University (UFF), Niterói, Brazil
marcosbedo@id.uff.br

Abstract. The handling of massive data requires the retrieval procedures to be aligned with the storage model. Similarity searching is an established paradigm for querying large datasets by content, in which data elements are compared by means of *metric* distance functions. Although several strategies have been proposed for the storage of data queried by metrics into relational schemas, no empirical assessment on the suitability of such strategies for similarity searching has been conducted. In this study, we aim at filling this gap by providing an in-depth evaluation of storage models for Relational Database Management Systems (RDBMS) in standard SQL. Accordingly, we propose a taxonomy, which divides approaches into four categories, *Binary*, *Relational*, *Object-Relational*, and *Semistructured*, and implement a representative storage model for each category within a common framework. We carried out extensive experiments on the four implemented strategies, and results indicate the *Relational* and *Object-Relational* storage models outperform the other competitors in most scenarios, whereas the *Binary* storage model reaches a good performance for queries with costly comparisons. Finally, the *Object-Relational* approach showed the best compromise between performance and representation, since its behavior is similar to the *Relational* storage model with a cleaner representation.

Keywords: Similarity searching · Metric data · RDBMS · SQL

This study has been supported by the Brazilian agencies CNPq, CAPES and Araucária Foundation under grants 426202/2016-3 and 88882.167843/2018-01.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 283–291, 2018.
https://doi.org/10.1007/978-3-319-98812-2_24

1 Introduction

The efficient handling of massive digital data requires the retrieval procedures to be aligned with the storage model. Traditionally, Relational Database Management Systems (RDBMS) are in charge of such tasks, as they include powerful routines for the searching of traditional data. However, the querying of more *complex data* requires the storage and search procedures of RDBMS to be extended. For instance, the comparison of images, movies, and time series by RDBMS Identity and Order operators is hardly useful for the retrieval of related elements [8]. Moreover, such data can be stored in multiple ways [5–7].

Complex data are commonly summarized as *feature vectors* (FVs), which consist of characteristics extracted from the original raw data. A FV can be seen as the result of the mapping of the contents of an element into a particular domain that is part of a *metric space*. Formally, a metric space $\mathcal{M}$ is a pair $\langle \mathbb{S}, \delta \rangle$, where $\mathbb{S}$ is the data space in which the elements are mapped, and $\delta : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}_+$ is a *metric* distance function (DF) that complies with the properties of non-negativity, symmetry, and triangle inequality [8]. In the context of metric spaces, the smaller the distance between two objects, the more similar they are to each other. Accordingly, RDBMS can employ DFs for the creation of new similarity searching operators for *content-based retrieval*. The first challenge for embedding such new operators in high-level SQL expressions lies in the finding of the most suitable approach for (i) the storage of original and mapped data, and (ii) the association of mapped characteristics with DFs.

Although several approaches have been proposed for enhancing the support of RDBMS towards similarity searching operators [1,4,5,7], an empirical assessment of techniques for complex data storage in relational schemas is yet to be conducted. In this study, we aim at filling this gap by providing an in-depth evaluation of storage models in standard SQL in terms of efficiency and usability. Accordingly, we propose a taxonomy, which divides existing approaches into four categories, namely *Binary*, *Relational*, *Object-Relational*, and *Semistructured*. We also design an extended version of the `FMI-SiR` framework [4] to implement a storage model for each category of our taxonomy. The extended framework, called *e*`FMI-SiR`, generically represents similarity searching operators by using high-level `CREATE FUNCTION` statements of SQL Data Definition Language (DDL). Accordingly, the contributions of this paper are:

1. **A taxonomy for complex data storage.** We propose a categorization of approaches regarding complex data storage on RDBMS using standard SQL resources.
2. **An empirical evaluation of data storage models.** We implement and evaluate a set of representative storage models in an extended framework based on standard SQL.

The remainder of this study is organized as follows. Section 2 discusses similarity searching and related work. Section 3 describes our approach. Section 4 presents experimental evaluations, while Sect. 5 concludes the paper.

2 Related Work

Previous studies discuss distinct solutions for FV storage. For instance, `SIREN` [1] is a middleware that can be coupled to an RDBMS. It defines two new data types, which are based on SQL/MM and proposes extensions to the SQL to enable these data types to be declared as the domain of complex attribute. A similar rationale is employed by `PostgreSQL-IE` and `SimDB`, which are specifically designed for PostgreSQL RDBMS. `PostgreSQL-IE` [3] stores complex data through a User-Defined Type (UDT) and employs a collection of User-Defined Functions (UDF) for the representation of similarity searching operators. `SimDB` [7] stores complex data as one table attribute and enables similarity-based retrieval by using a modified SQL that supports `JOIN` and `GROUP BY` similarity operators. Frameworks `MSQL`, `MESSIF`, and `FMI-SiR` follow a different premise for storing complex data. `MSQL` [5] explicitly stores the data and features as attributes based on user-defined relations, whereas `MESSIF` [2] stores complex data features in semi-structured objects (buckets) that represent partitions of the search space. Finally, `FMI-SiR` [4] stores the complex data and their features in two binary attributes.

Besides logical and physical differences in the storage of complex data, existing approaches also diverge in the way they represent similarity searching operators. `SIREN`, `MESSIF` and `SimDB` use their own extended SQL compilers for query interpretation. In contrast, `MSQL` and `FMI-SiR` benefit from the RDBMS architecture itself, which allows the extension of operators by using only standard SQL `CREATE FUNCTION` statements. Such a characteristic enables the RDBMS to apply standard query optimization procedures. In this context, `FMI-SiR` uses the extensible interfaces offered by Oracle Data Cartridge for the binding of external C/C++ functions of the Arboretum library[1]. Analogously, `MSQL` defines similarity searching operators by using a UDF called `DIST` for the filtering of elements according to the criteria defined in other UDF called `LOCATE`.

3 Storage/Retrieval of Complex Data in Standard SQL

We propose an extension to `FMI-SiR` framework on top of Oracle RDBMS, called *e*`FMI-SiR`, for the implementation of existing complex data storage/retrieval approaches by *using only standard SQL statements*. Thus, users can choose the format of complex data storage. We argue DFs are special constructors that enable the query processor to identify similarity operators for any query. Accordingly, we propose to implement the constructors by imposing the `DISTANCE` modifier to `CREATE FUNCTION` statements. With such a simple extension, UDF blocks can be recognizable by the RDBMS compiler. Therefore, the main types of similarity-based queries can be represented using standard SQL, which are not show in the paper due to space limitations, and, moreover, they can be unambiguously detected for optimization purposes.

[1] https://bitbucket.org/gbdi/arboretum.

3.1 Representation of Distinct RDBMS Data Models

Key issues for the handling of complex data storage are (i) how FVs are stored, and (ii) how to represent DFs within similarity searching regardless of the data model. Our proposal takes into account the following possibilities:

1. *Relational:* FVs are stored in system tables of the RDBMS catalog and DFs are in procedural SQL;
2. *Binary:* FVs are stored in `BLOB` columns and are manipulated through external functions;
3. *Object-relational:* FVs/DFs use object-relational RDBMS functionalities;
4. *Semi-structured:* FVs are handled as semi-structured data and DFs use parser APIs.

Relational Data Model. Feature vectors with a fixed number of features can be stored into a relation, and indexes can be defined by using the multi-column syntax. However, FVs with a varying number of features may produce a large number of attributes for the covering of special cases, which are `NULL` in the average case. A major drawback of this approach is FVs are described by *table structures* and not by the *complex data type*, which may be not suitable for the definition of the DF parameters. For instance, the parameters cannot be the *table name*, as this option would not be applied to out-of-row data (e.g., a query element that is not in the database). The parameters also cannot be a general `RECORD` or even a `CURSOR` because it prevents a proper DF-based type checking. Therefore, DF parameters must be the individual FV features. Since each feature is bounded by a distinct parameter, and parameters are handled by name, it is necessary the creation of a DF instance for every particular case, *e.g.*, the dimensionality of the FV.

Binary Data Model. RDBMS are unaware of data content for domain-specific solutions. The management relies on the storage of binary data into `BLOB` columns of the data table, whereas the columns content are manipulated by external functions. Therefore, it is also required the inclusion of metadata into the FV structure. Such an approach does not take advantage of RDBMS type checking and also demands the manipulation of data through `LOB` functions or external function calls, which may lead to access overhead regarding every FV. The index creation follows the usual single-attribute syntax.

Object-Relational Data Model. The Object-Relational Data Model allows the definition of a proper type hierarchy for FVs and static DFs. There are two options for storing features in an FV type: (i) as member attributes, or (ii) using arrays. The first option stores FVs either as single object tables or as nested tables according to the object definition. Unfortunately, such option requires the creation of non-generic data types and also demand the creation of specific DF for every dimensionality. On the other hand, *arrays* provide a generic

and flexible representation in standard SQL. Arrays are length adjustable, which enable them to store FVs with either fixed or variable number of features, as well as they enable fast iteration. Accordingly, we use arrays as object-relational representations. In our approach the index creation also follows the single-attribute syntax.

Semi-structured Data Model in Standard SQL. Semi-structured data has been widely employed for many applications, being XML and JSON the most used data types. We employ XML in this study because its support in standard SQL is mature. The whole amount of FV types can be uniformly represented using a single semi-structured data type, in this case, an XMLType. Constraints that should be enforced for FVs can also be stated in an XML Schema. Both parameter binding and index creation are similar to the binary approach, but with the improved type and integrity checking. XML engines in RDBMS usually support different storage options. Oracle Database, for instance, enables the storage of an XMLType using either a BLOB that is a serialized version of the XML data or using shredding, which extracts data in a relational format and stores them in system tables. Hereafter, we refer to these two approaches as XML Binary and XML, respectively.

4 Experiments

This section reports on an experimental evaluation of the four reviewed strategies for the storage and retrieval of complex data. In particular, we evaluate all these strategies over the real and public `CoPhIR`[2] (**C**ontent-based **P**hoto **I**mage **R**etrieval) dataset. We experiment on three distinct DFs, two of Minkowski family L_1 and L_2, and function Mahalanobis. In the first experiment, we measured the total amount of space required for the storage of `CoPhIR` elements by using all implemented storage data types. The second experiment reports on the demanded time of insertions for an increasing number of dimensions. Our last experiment measures the total time spent in the execution of sequential scan range queries for a varying number of dimensionalities and cardinalities.

We defined five implementations for FV storage: *Relational*, *Object-Relational*, which stores the dimensions as *VArray*, Semistructured, which stores data as either *XML-Binary* files or shredding *XML*, and *Binary*, which stores FVs as `BLOB`. All experiments were performed on a computer with an Intel(R) Xeon(R) CPU E5-2420 0 @ 1.90 GHz processor, cache size of 15 MB, RAM memory size of 8 GB, Ubuntu 14.04 OS and RDBMS Oracle 12c.

4.1 Complex Data Storage Evaluation

We measured the overall disk space required for the storage of `CoPhIR` entries. First, we vary the size of each FV and annotated the impact of *dimensionality* for

[2] http://cophir.isti.cnr.it/.

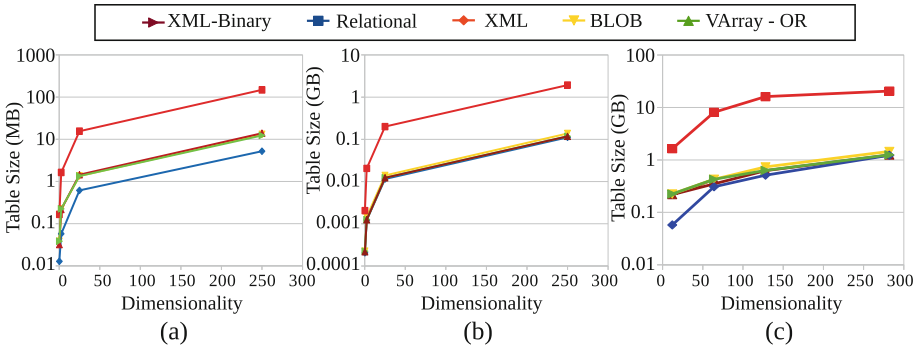


Fig. 1. Storage space for (a) 10K, (b) 100K, and (c) 1M FVs.

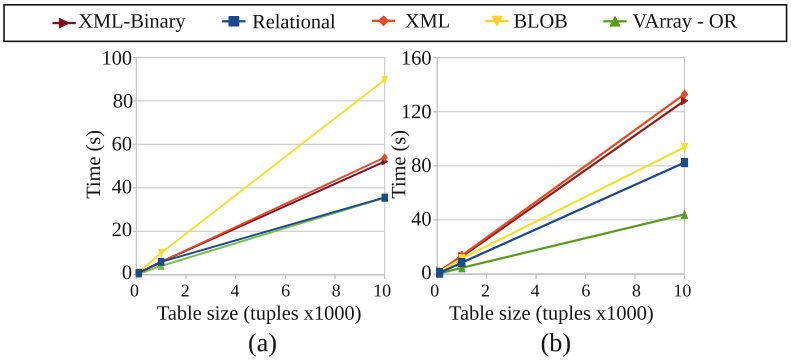


Fig. 2. Insertion of complex data. (a) 12 and (b) 282-dimensional FVs.

values of 12, 64, and 282 dimensions, which corresponds to the `CoPhIR` dimensions that represent color, texture, and shape. Figure 1 shows the disk space demanded by each storage model. Shredded XML required more disk space than the competitors in all evaluated cases. In particular, it was up to 25x more disk-consuming than the XML-Binary approach regarding the dataset with one million tuples. Although both Relational, VArray, Binary, and XML-Binary have demanded similar disk space, the Relational approach was slightly better than the other competitors, especially for the 12-dimensional scenario in which it was up to 57% less disk-consuming than the closest competitor, VArray.

4.2 Complex Data Insertion Evaluation

As for the evaluation of transactional operations on complex data, we measured the time demanded to insert FVs. In this experiment, we measured the accumulated time spent in the insertion of an increasing number of tuples. Figure 2 shows the accumulated times regarding `CoPhIR` FVs of 12 and 282 dimensions. Results indicate insertion times grow linearly with increasing number of tuples.

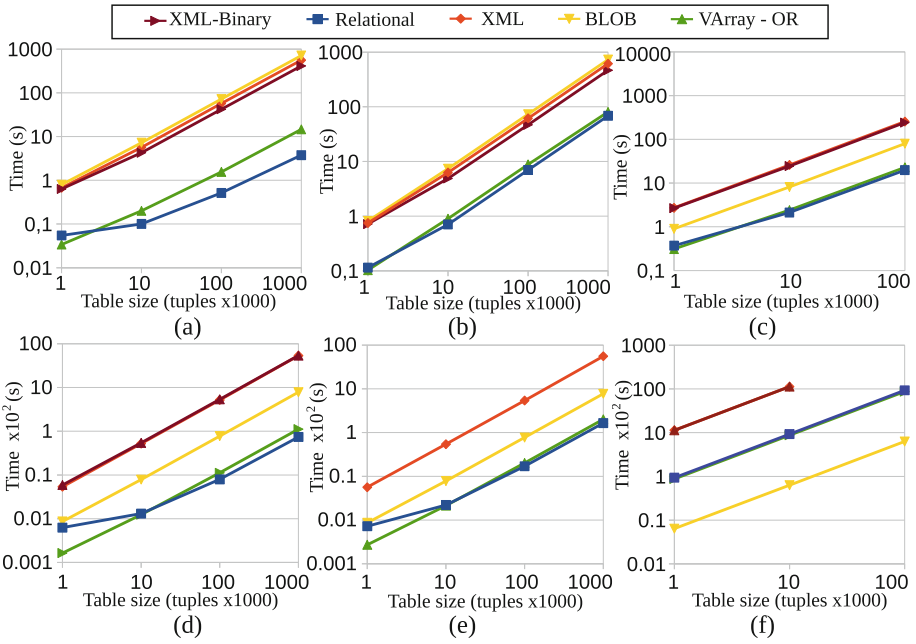


Fig. 3. Range queries for retrieval of at most 1% of the tuples. (a–c) 12-dimensional FVs queried by L_1, L_2, and Mahalanobis DF, respectively. (d–f) 282-dimensional FVs queried by L_1, L_2, and Mahalanobis DF, respectively.

In addition, Binary and VArray insertion were not sensitive to the number of dimensions, while other models revealed to be very sensitive. For instance, Relational storage was 130% slower to insert a 282-dimensional FV in comparison to its performance regarding the insertion of a 12-dimensional FV, being up to twice slower than VArray. Likewise, XML-based storage models also were be affected by dimensionality and the time spent in the insertion of 282-dimensional vectors was up to 133% slower in comparison to the 12-dimensional case.

4.3 Complex Data Retrieval Evaluation

We measured the impact of FV storage in the execution of similarity searching by executing range queries which retrieve up to 1% of the stored FVs. We defined 100 elements of the entire `CoPhIR` database to be used as query elements and remove them from the queried tables. Figure 3 reports on the average time demanded by the execution of a range query on a varying number of dimensions.

Relational storage achieved the fastest retrieval of complex data for 12-dimensions (Fig. 3(a–c)), regardless of DF. VArray was the closest competitor and performed almost as fast, except for the L_1 case (Fig. 3(a)). The performance differences between the storage approaches is more evident for the 282-dimensional FVs (Fig. 3(d–f)). In such cases, XML storage, either shredded or

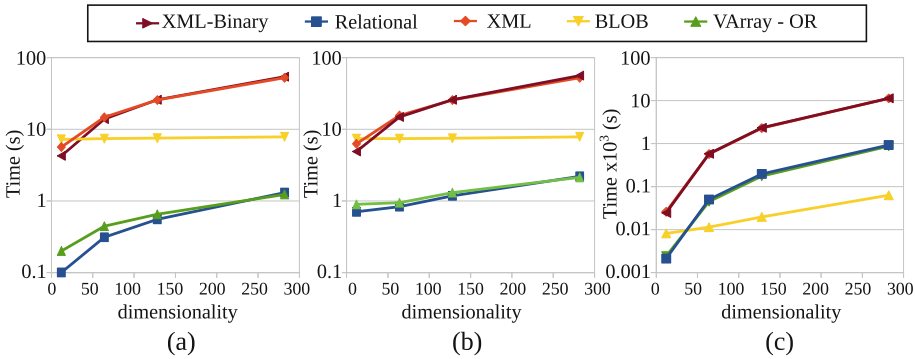


Fig. 4. Dimensionality impact for queries: (a) L_1, (b) L_2 and (c) Mahalanobis.

as a binary file, was clearly slower than the others due to the XML DOM API overhead. No difference was observed between VArray and Relational storage, which were the most suitable approaches for data retrieval for both L_1 and L_2 comparisons. In contrast, Binary approach was by far the fastest approach for the Mahalanobis distance comparisons for large dimensionalities (Fig. 3(f)).

We performed a second evaluation to inspect this behavior closely and queried `CoPhIR` with an increasing number of dimensions and the fixed cardinality of 10,000 tuples. Figure 4 shows Relational and VArray were consistently faster than BLOB and XML for L_1 and L_2, while Binary storage was always more efficient than them for Mahalanobis DF and dimensionalities larger than 64. Such a result reinforces distinct storage data models impact differently in the execution time of queries issued through standard SQL.

5 Conclusions

Previous studies extended RDBMS capabilities for the handling of similarity searching operators. However, no empirical assessment on the suitability of such approaches for complex data manipulation had been conducted. We have filled this gap by (i) defining a taxonomy of approaches using standard SQL, and (ii) providing an in-depth evaluation of RDBMS storage models. Experiments showed *Relational* storage model outperforms the competitors in most scenarios, whereas the *Binary* model performs better for queries that employ costly distance-based comparisons. Finally, the *Object-Relational* approach reached the best compromise between performance and representation, since its behavior is similar to the *Relational* strategy with a cleaner representation. Therefore, we recommend the implementation of this last data storage model in future works.

References

1. Barioni, M., Razente, H., Traina, A., Traina-Jr., C.: SIREN: a similarity retrieval engine for complex data. In: PVLDB, pp. 1155–1158. VLDB Endow (2006)
2. Batko, M., Novak, D., Zezula, P.: MESSIF: metric similarity search implementation framework. In: Thanos, C., Borri, F., Candela, L. (eds.) DELOS 2007. LNCS, vol. 4877, pp. 1–10. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77088-6_1
3. Guliato, D., Melo, E.V., Rangayyan, R.M., Soares, R.C.: PostgreSQL-IE: an image-handling extension for PostgreSQL. J. Digit. Imaging **22**(2), 149–165 (2009)
4. Kaster, D.S., Bugatti, P.H., Traina, A.J.M., Traina-Jr., C.: FMI-SiR: a flexible and efficient module for similarity searching on Oracle database. J. Inf. Data Manag. **1**(2), 229 (2010)
5. Lu, W., Hou, J., Yan, Y., Zhang, M., Du, X., Moscibroda, T.: MSQL: efficient similarity search in metric spaces using SQL. VLDB J. **26**(6), 829–854 (2017)
6. Shimura, T., Yoshikawa, M., Uemura, S.: Storage and retrieval of XML documents using object-relational databases. In: Bench-Capon, T.J.M., Soda, G., Tjoa, A.M. (eds.) DEXA 1999. LNCS, vol. 1677, pp. 206–217. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48309-8_19
7. Silva, Y.N., Aly, A.M., Aref, W.G., Larson, P.A.: SimDB: a similarity-aware database system. In: SIGMOD, pp. 1243–1246. ACM (2010)
8. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, 1st edn. Springer, New York (2010). https://doi.org/10.1007/0-387-29151-2

A Native Operator for Process Discovery

Alifah Syamsiyah(✉), Boudewijn F. van Dongen, and Remco M. Dijkman

Eindhoven University of Technology, Eindhoven, The Netherlands
{A.Syamsiyah,B.F.v.Dongen,R.M.Dijkman}@tue.nl

Abstract. The goal of process mining is to gain insights into operational processes through the analysis of events recorded by information systems. Typically, this is done in three phases. Firstly, events are extracted from a data store into an event log. Secondly, an intermediate structure is built in memory and finally, this intermediate structure is converted into a process model or other analysis results.

In this paper, we propose a native SQL operator for direct process discovery on relational databases. We merge steps 1 and 2 by defining a native operator for the simplest form of the intermediate structure, called the "directly follows relation". We evaluate our work on big event data and the experimental results show that it performs faster than the state-of-the-art of database approaches.

Keywords: SQL operator · Relational database · Process discovery

1 Introduction

Process mining is a research discipline that turns event data into process models, checks the model with reality, and enhances the model with statistics derived from event data. There has been an extensive research in process mining, including *process discovery, conformance checking*, and *enhancement*.

In the current state-of-the-art, process mining tools need event logs as input. Such an event log consists of events pertaining to who executed which activity at what point in time for which case. One of the characteristics of event logs is that they are static, i.e. one file only contains one case notion and contains data for one specific period of time.

Such event logs are typically extracted from databases. The event log is then loaded into a process mining tool which builds an in-memory abstraction for further processing. Given the fact that many legacy information systems use a database as the back end, we study the question how to build such an abstraction directly in the database [2,9–11]. In this paper, we focus on process discovery, i.e. we focus on deriving a process model from event data. In [9], we showed how to compute one of the most relevant abstractions, namely the directly follows relation (DFR), inside a database and how to only import the structure (not the event data) into a process mining tool. The mining tool then discovers the process model using an existing algorithm.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 292–300, 2018.
https://doi.org/10.1007/978-3-319-98812-2_25

When data is stored in an SQL database, it is rather trivial to obtain the data from the database through a so-called SQL interface. By simply ordering the events by case and then time and retrieving the entire set of events. Unfortunately, this leads to massive communication between a process mining tool and the database and still requires the process mining tool to build the abstraction in memory on the result of this query.

In order to compute the abstraction in database, [9] proposes the use of nested SQL queries, which, unfortunately, are not designed towards process mining purposes.

Therefore, in this paper, we propose a native SQL operator for direct process discovery in relational databases, which is designed for a specific process discovery purpose. As a starting point, this paper investigates the directly follows relation. However, we do not restrict the possibility to extend the operator to other kinds of abstractions. Using this native operator, the database has more flexibility to query process mining related questions. Moreover, it harnesses advances in database technology to speed up the computation time. Figure 1 highlights the four different approaches mentioned above.

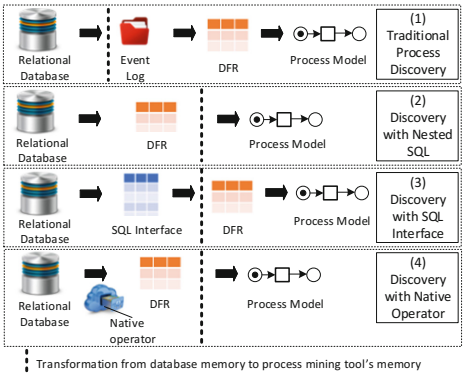


Fig. 1. Four different approaches to process discovery

The reminder of this paper is structured as follows. Section 2 discusses some related work including traditional process discovery techniques. In Sect. 3 we examine in database approaches for process discovery. Section 4 demonstrates the experimental results and finally the paper is concluded in Sect. 5.

2 Related Work

Abstractions play an important role in process discovery. The work in [9], investigates directly follows relations and declarative relations as abstractions. Another example is the work in [10] which explores handover of work abstraction for social network analysis. Both approaches are built on top of the SQL basic syntaxes which are not specialized for process discovery purposes. As a result, those techniques are not versatile for processing big event data.

Indeed, SQL – as the lingua franca for relational database systems – can no longer be considered to meet all requirements of modern applications. More and more technologies with built-in set of natively implemented functions have appeared, e.g. SAP HANA [4], QUEST [1], DBMiner [5], and KnowledgeMiner [7].

Within the context of process mining, process discovery deals with the extraction of a process model from an event log, such that the model is representative

for the behaviour seen in the event log [12]. A large variety of process discovery algorithms have been proposed, including Alpha Miner [13], Inductive Miner [6], and many others.

While the details may differ, all process discovery techniques have three phases in common: (1) *loading*, (2) *abstraction*, and (3) *mining*. *Loading* is the phase during which an event log is imported into a process mining tool. Often the logs are stored as CSV files or XES event log files. *Abstraction* is the phase during which an event log is transformed into a more compact data structure. For example, for the Alpha Miner, this abstraction is the DFR on activities (denoted by $A >_L B$), which holds for two activities A and B if and only if somewhere in the event log L, there are two successive events in a trace corresponding to these activities. Finally, based on this abstraction, during the *mining* phase a process model is discovered.

The state-of-the-art in process discovery at this point is the Inductive Miner. This miner also uses an abstraction in terms of a DFR, but rather than the Alpha Miner where only the occurrence of direct succession is relevant, for the Inductive Miner, the frequency with which two activities directly succeed each other is counted.

Our paper builds on the work in [3] which provides a theoretical foundation for the native directly follows operator in an SQL database. In our paper, we put the native operator into process mining contexts, in particular process discovery and we merge the native operator with the state-of-the-art of process discovery, namely the Inductive Miner, to enable real in-database process mining. We focus on the DFR as an abstraction since it is used in common process discovery algorithms.

3 Process Discovery on Relational Databases

The key idea of direct process discovery on relational databases is that we skip the loading phase and we perform the abstraction phase inside the database management system. Rather than exporting event data to a file, we execute a query for getting the abstractions.

3.1 Nested SQL for Directly Follows Relation

Let us consider a simple log stored in Table *Log* (see Table 1a). Using a standard query language in database (SQL), we compute the DFR from the Table *Log* as follows:

```
1  SELECT a.Activity, b.Activity, count(*) FROM Log a, Log b
2  WHERE  a.Case = b.Case AND a.Time < b.Time AND
3  NOT EXISTS (SELECT * FROM Log c WHERE c.Case = a.Case
4              AND a.Time < c.Time AND c.Time < b.Time)
5  GROUP BY a.Activity, b.Activity;
```

In the query above, we need an inner join between two instances of Table *Log* to acquire pairs of activities. Afterwards we check whether the two activities from each pair come from the same case. If so, we take two consecutive activities

Table 1. (a) Example of event data stored in Table *Log*, (b) The DFR of Table *Log*

Case	Activity	Time
1	Send request	2017/10/01 08:45:30
1	Check application	2017/10/02 10:34:14
1	Accept	2017/10/05 12:13:24
2	Send request	2017/10/03 16:31:16
2	Check application	2017/10/07 15:17:36

Event_Label_P	Event_Label_S	Frequency
Send request	Check application	2
Check application	Accept	1

a and b if there is no other activity c between them (here we need a negative query). Finally we return the pairs of activities and their frequency by grouping the same pairs. The result of this query is denoted in Table 1b.

The query illustrates the complexity of computing a very basic process mining relation using standard SQL syntax. The nested query (i.e. the "NOT EXISTS" part) causes performance problems in any database system. In [3] it has been proven that executing a nested query to compute the DFR leads to third order polynomial costs and this is also shown in our experiments in Sect. 4.

To avoid the problem of inefficiency, one might be tempted to define a query without a "NOT EXISTS" part, which exploits a time-order of the events to derive the weakly follows relation. For example, assuming that the database management system can return the row number of each tuple, a relation can be ordered by case identifier and time, and subsequently the following SQL query would return the directly follows relation for event logs in which no events occur at the same time.

```
1  SELECT DISTINCT a.Activity, b.Activity, count(*)
2  FROM Log a, Log b WHERE a.Case = b.Case AND
3  a.ROW_NUM = b.ROW_NUM - 1 GROUP BY a.Activity, b.Activity;
```

However, such a query fails if two events happen at the same time. For example, for the log $(1, A, 0{:}01), (1, B, 0{:}02), (1, C, 0{:}02), (1, D, 0{:}03)$, this query would not work, because the query would return $(A, B), (B, C), (C, D)$, while it should return $(A, B), (A, C), (B, D), (C, D)$. What makes matters worse, is that the result would be non-deterministic, since the result that would be returned, depends on the particular order of events that happen at the same time, which is not specified. This problem may be more or less common in a log, depending on the time-granularity at which events are logged, which – in practice – is often (too) coarse. For example, it may happen that only the date of the event is logged and not the time. This problem is also known as the *timestamp challenge* [14].

3.2 SQL Interface

One way to circumvent the complexity of nested queries is to split the computation of the directly follows relation in two parts: (1) sorting the log based on cases and timestamps in the database, and (2) counting the frequency for each two consecutive activities in the process mining tool.

As denoted in Fig. 1c, part (1) can be translated into an SQL query as follows:

```
1  SELECT * FROM Log ORDER BY Case, Time;
```

This query is executed through an SQL interface and the result is then transferred to the process mining tool. Inside the process mining tool, the DFR is computed in the sorted event log. The downside of this approach is that the whole log needs to be transferred from the database to the process mining tool.

3.3 Native Directly Follows Operator

The nested form of the earlier SQL query emphasizes that the SQL syntax is not suitable for computing the DFR. Therefore, we propose a native operator `directlyfollows`, which requires a table object representing the event log consisting of three columns: the case, the activity, and the timestamp. It returns a table object representing the DFR, which again consists of three columns: the first and the second activity in the relation, and the frequency of each pair.

Assume that we have a Table *Log* (see Table 1a), we compute the DFR using the native operator `directlyfollows` as follows.

```
1  SELECT * FROM DIRECTLYFOLLOWS (SELECT * FROM Log);
```

After the database engine parses `directlyfollows`, it calls a function which reads the log, then sorts it by case identifiers and timestamps and finally returns pairs of consecutive events and the frequency (as denoted in Table 1b). The complexity is worst case $|E| \log |E|$ (with E is the total number of events) due to the required sorting of events. The technical details of such function can be found in [8].

In the context of process discovery, the DFR is then retrieved by a process mining tool which uses this to produce further analysis results. Note that the result from the `directlyfollows` operator can be used directly in the existing process discovery technique without modifying the algorithm or reinventing a discovery algorithm.

Process discovery using the native operator has several advantages. (1) The query can be expressed in a straightforward way, hence it is more convenient for novice users to express various process mining related questions. (2) The query outperforms other database techniques (see Sect. 4). (3) The abstraction is built inside the relational databases, thus saving memory in the process mining tool. (4) There is no need to extract and load a log file into a process mining tool, thus saving time. (5) The DFR can be computed upon insertion of data using the standard triggering mechanism of any database system, hence eliminating the need for the process analyst to wait for the computation to finish.

4 Experimental Results

We implemented our work in ProM and H2 and we compared the native operator (Fig. 1d) with the other three approaches: (1) the traditional technique (Fig. 1a), (2) the discovery with Nested SQL (Fig. 1b), and (3) the SQL interface (Fig. 1c). The starting point for the traditional approach is an XES event log file already loaded into memory. For the database approaches, the starting point

is a database in which events have been inserted[1]. In the initial experiments, we did not use any type of indices for the events so we can represent general scenarios where table logs are constructed without any indices. However, in the subsequent experiments, we also investigate scenarios where indices are utilized.

We created two kinds of synthetic logs: (a) logs with an increasing number of activities, and (b) logs with an increasing number of events. For (a), we relabelled activities to vary the number of activities between 30 to 3840 while keeping the same number of events. For (b), we merged one case with another case to vary the number of events between 1 K to 42M while keeping the same number of activities. This way, we preserve the control flow for our extended logs in the same way as the control flow of the original log.

Figure 2 shows the time for the abstraction phase of the four approaches. On the left, the time is shown as a function of the number of events in the log and on the right as a function of the number of activities. As expected, the time complexity of the native, SQL interface, and traditional approaches is linear because events in the log are already sorted. However, we cannot see the last dot of the traditional approach. This is because the traditional approach cannot handle the biggest log containing 42M events due to out of memory exception. Furthermore, the execution of the nested query leads to third order polynomial time complexity.

The right hand side of Fig. 2 shows that the number of activities does not affect the performance of the database approaches[2]. However, for the traditional approach, there is an influence due to the fact that, while scanning the log, internal data structures need to grow to accommodate for previously unseen activities. For <1000 labels, the time is consistently around 30 s. For 1920 activities, the time is around 70 s and for 3840 activities, the time grows to 409 s. If the number of activities is known upfront, the implementation could try to allocate sufficient memory upfront, thus eliminating this effect.

For the database approaches, the pre-processed data needs to be retrieved from the database into the process mining tool. As shown in Fig. 3, the retrieval phase in SQL interface is linear in the number of events, since each event is transferred. In contrast, the line of native operator is flat, since the events themselves are not retrieved, but only the non-zero elements in the directly follows relation. This is shown by the right-hand figure, showing the influence of the number of activities in retrieval phase. Both native and SQL interface demonstrate a second order polynomial time complexity since retrieving the DFR is (worst case) quadratic in the number of activities.

The time complexity of Inductive Miner is worst-case cubic in the number of activities, which is clearly shown in Fig. 4. There is no difference between the four approaches since they use the same implementation of the Inductive Miner.

[1] We used an H2 database server with 64 GB of RAM and 8 CPU cores@2.40 Ghz. Discovery was done on a PC with 8 GB of RAM and 2 CPU cores@2.30 Ghz.

[2] Due to the fact that the nested query is so time-consuming, we did not include it in some of the tests.

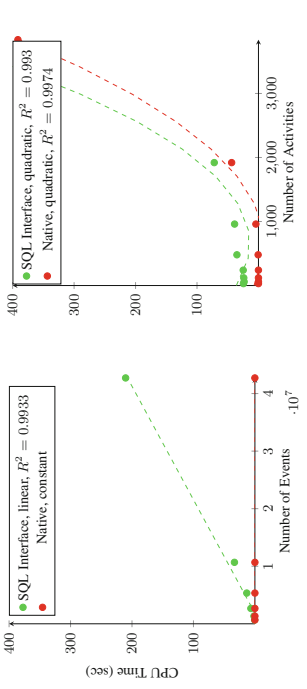


Fig. 2. CPU time vs. events and activities in the abstraction phase.

Fig. 3. CPU time vs. events and activities in the retrieval phase.

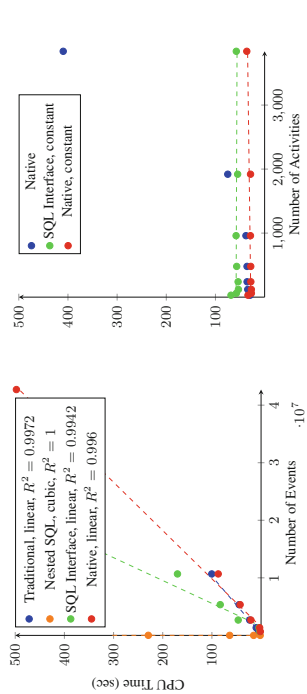


Fig. 4. CPU time in the mining phase.

Fig. 5. Effect of indexing on CPU time.

Fig. 6. Total CPU time for datasets with increasing number of events or increasing number of activities.

Figure 6 denotes the total time of Figs. 2, 3 and 4, i.e. the total time of abstraction, retrieval, and mining phases. In Fig. 6, we do not include the nested query because it is so time-consuming (the experiments with nested query were stopped until the number of events reached 14K). However, we still tried to improve the performance of nested query by adding indices in case and timestamp columns. Even though the cubic complexity in the nested query is reduced to $x \cdot \log x$ (with x is the total number of events), the native approach still outperforms the nested query because the former has linear complexity as shown in Fig. 5[3].

From Figs. 5 and 6 it becomes clear that for large numbers of events and/or large numbers of activities, the native operator outperforms the other database approaches. Both the native and traditional approaches show relatively similar performance, except for the log with the largest number of events (the 6^{th} log in the left chart of Fig. 6) for which the traditional approach runs out of memory.

5 Conclusion

In this paper we focus on direct process discovery on relational databases. We propose a native SQL operator for directly computing the Directly Follows Relation (DFR).

The evidence presented in this paper has shown that the native operator allows to express the query to obtain the DFR in a straightforward way, hence it supports multi perspective in process mining. Moreover, the native operator has been implemented in H2 database and we provide a specialized implementation of the state-of-the-art process mining technology to show applicability of the work.

Using experiments, we compare various techniques to do process mining on relational databases and we show our query outperformes them in the context of big event data.

For future work, we plan to include a time perspective in the discovery. Based on the time t defined in the user's query, we will turn on a database trigger to automatically discover a process model with frequency f. Furthermore, we also aim to enable a direct conformance checking in the database.

References

1. Agrawal, R., Mehta, M., Shafer, J., Srikant, R., Arning, A., Bollinger, T.: The quest data mining system. In: KDD 1996, pp. 244–249. AAAI Press (1996)
2. Calvanese, D., Montali, M., Syamsiyah, A., van der Aalst, W.M.P.: Ontology-driven extraction of event logs from relational databases. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 140–153. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42887-1_12

[3] Note that the linearithmic comes from the fact that H2 database uses B-tree index, hence finding an element is $\mathcal{O}(\log x)$. There are x rows for which we need to perform this look up, therefore the complexity is $\mathcal{O}(x \cdot \log x)$.

3. Dijkman, R., Gao, J., Grefen, P., ter Hofstede, A.: Relational algebra for in-database process mining (2017)
4. Färber, F., Cha, S.K., Primsch, J., Bornhövd, C., Sigg, S., Lehner, W.: SAP HANA database: data management for modern business applications. SIGMOD Rec. **40**(4), 45–51 (2012)
5. Han, J., Cai, Y., Cercone, N.: Data-driven discovery of quantitative rules in relational databases. TKDE **5**(1), 29–40 (1993)
6. Leemans, S.J.J.: Robust process mining with guarantees. Ph.D. thesis, TU Eindhoven (2017)
7. Shen, W., Ong, K., Mitbander, B., Zaniolo, C.: Metaqueries for data mining (1996)
8. Syamsiyah, A., van Dongen, B.F., Dijkman, R.: Native directly follows operator. CoRR, abs/1806.01657 (2018)
9. Syamsiyah, A., van Dongen, B.F., van der Aalst, W.M.P.: DB-XES: enabling process mining in the large. In: SIMPDA 2016 - Extended Versions, pp. 63–77 (2016)
10. Syamsiyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Discovering social networks instantly: moving process mining computations to the database and data entry time. In: Reinhartz-Berger, I., Gulden, J., Nurcan, S., Guédria, W., Bera, P. (eds.) BPMDS/EMMSAD -2017. LNBIP, vol. 287, pp. 51–67. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59466-8_4
11. Syamsiyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Recurrent process mining on procedural and declarative approaches. BPM Center Report BPM-17-03 (2017)
12. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
13. van der Aalst, W.M.P., Weijter, A.J.M.M., Maruster, L.: Workflow mining: discovering process models from event logs. TKDE **16**, 1128–1142 (2004)
14. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28108-2_19

Implementation of the Aggregated R-Tree for Phase Change Memory

Maciej Jurga and Wojciech Macyna(✉)

Department of Computer Science, Faculty of Fundamental Problems of Technology, Wrocław University of Technology, Wrocław, Poland
maciej.dariusz.jurga@gmail.com, wojciech.macyna@pwr.wroc.pl

Abstract. Phase change memory (PCM) has become one of the most promising non-volatile memory type. To make the storage of spatial objects PCM friendly, the traditional R-tree index must be implemented from scratch in order to take into account the characteristics of PCM. The aggregated R-tree extends the original R-tree with the aggregated values connected with the nodes.

In this paper, we deal with the storage technique for aggregated values of the R-tree index optimized for PCM. The proposed Aggregation R-tree Method (ARM) records the most profitable set of the aggregated values in the case when the memory size is limited. The method chooses to store frequently asked values and the aggregated values with high calculation cost. The replacement technique used in the method requires three times fewer write operations in comparison to FIFO (First In First Out) for read-heavy workload.

Keywords: Phase change memory · Aggregated value · R-tree
Spatial database

1 Introduction

Phase change memory (PCM) has become one of the most promising non-volatile storage media. Due to its byte addressability, high access speed and low energy consumption, PCM is the best candidate to cooperate with DRAM as a main memory storage type. However, the drawbacks of PCM are: the limited write endurance and high write latency. PCM can endure only 10^7–10^8 writes, what makes it impossible to completely replace DRAM. Therefore, it is reasonable to consider a hybrid main memory system, which consists of both: PCM and (a relatively small amount of) DRAM. Nowadays, the research of PCM is concentrated on many aspects. To prolong the lifetime of PCM, we need a mechanism that distributes the write operations to PCM cells uniformly (see [1–3]). Consequently, many data management techniques must be changed in order to reduce the write traffic. In [4], the authors deal with B+ tree index optimized for PCM. In [5], sort and join algorithms are considered. The paper [6] proposes the optimization of SQLlite with the PCM for mobile application. At last, in [7] the R-tree is proposed.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 301–309, 2018.
https://doi.org/10.1007/978-3-319-98812-2_26

The growing amount of spatial and geographical data requires efficient storage methods. Fast access to such information is possible only if the proper indexes are used. R-tree is one of the most popular spatial index. It splits the spatial area into rectangle-shaped regions. Every entry in the R-tree corresponds to the rectangle of the considered area and has pointers to the smaller rectangles inside it (see [8]). Figure 1 shows an example of the searching area R, which consists of three main subareas: A, B, C. The subareas are further split into smaller ones. The R-tree index for this area is reflected on Fig. 2.

The aggregated R-tree (aR-tree) index may be treated as an extension of the standard R-tree (see [9]). In that index, the aggregated values connected with the R-tree node are explicitly stored, so they do not need to be processed with each query, but may be fetched from the memory cache. For example, we can calculate the number of the objects in A based on the number of objects in $A1$, $A2$, $A3$ and then memorize this value inside the R-tree for the future use. Clearly, there may be many aggregated values assigned to one R-tree area. Let's suppose a situation that the city map containing many different spatial objects is indexed by the R-tree. The objects may be categorized. Each object may have many attributes, etc. It would be problematic to store all of them, particularly in the systems with limited memory capacity. On the other hand, the calculation of the aggregated value every time from scratch would increase the time and energy cost. Therefore, the balance between materialization and calculation is needed.

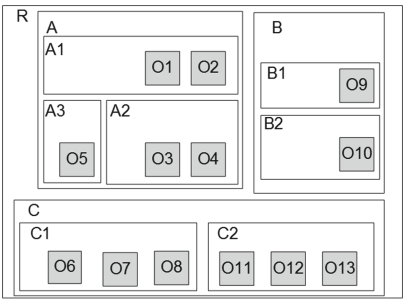


Fig. 1. Indexed area

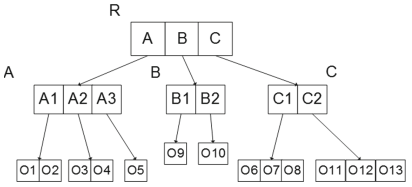


Fig. 2. R-tree for the indexed area

The **contribution** of this paper can be summarized as follows. We propose an implementation of the aggregated R-tree index optimized for phase change memory. To deal with memory limitation, we invent an algorithm that chooses the most profitable aggregated values to store. Our approach clusters the aggregated values into different partitions according to their calculation cost. To increase the overall performance, the values with high calculation cost are preferred. We perform several experiments to prove the efficiency of the method.

2 Aggregation R-Tree Method for PCM

In this section, we provide the Aggregation R-tree Method (ARM) for storing the aggregated values of the R-tree for PCM. We assume that the aggregated values are stored in RAM and Aggregation Cache (AC) which is located on PCM.

2.1 Query Evaluation Cost

Our method considers the calculation cost connected with the aggregated value. It is strongly correlated with the memory access cost.

Let r and r_L be the approximate reading cost of one aggregated value from the Aggregation Cache and the R-tree leaf node, respectively. The calculation cost $c(N)$ for the node N can be defined as follows:

$$c(N) = \begin{cases} 0; & \text{if } v_N \in \text{RAM} \\ r; & \text{if } v_N \in \text{AC} \\ r_L; & \text{if } N \in R_{leafnodes} \\ \sum_j c(N_j) \text{ for each } N_j \in N_{children}; & otherwise \end{cases}$$

2.2 System Architecture

In our method, we store the following data:

- **R-tree.** It facilitates spatial access to the data. The tree has height h which denotes the number of levels (without leaves). By f_{min} and f_{max}, we denote the minimal and maximal number of children per node, respectively. Thus, every R-tree node consists of e entries, where $f_{min} \leq e \leq f_{max}$.
- **Object types.** Different types of data may be connected with an R-tree node (e.g. "buildings", "trees" or "cars"). Let O be a set of objects stored in the system. By o_N we will denote the set of aggregated values calculated for node N.
- **Aggregated types.** Aggregation functions like: *min*, *max*, *count* and so on.
- **Aggregated values.** The aggregate value is depicted as $v_{N,o,s}$, where N is a node number, o an object type and s is an aggregate type. An example of the aggregated value may be the minimal height of buildings in the node area.

The architecture of the method consists of the following parts:

- **R-tree storage.** It stores the R-tree data.
- **Aggregation Cache.** The PCM area which has a fixed size and holds aggregated values. It consists of partitions. Each partition contains memory buckets. The buckets form a list that can be extended when a new aggregated value is added. There is one bucket list per partition and there are $h - 1$ partitions in total: one for each tree level except the root which belongs to the partition $h - 1$.

- **RAM cache.** It is used to store calculated aggregate values. This cache is searched before other structures. RAM cache follows the LRU (Least Recently Used) policy. In this way, we favor the values recently used.
- **PCM wear leveling system.** It is used for all data stored on PCM to map logical and physical addresses. This is not discussed in this paper (see [1–3]).

Our method works as follows. After request for an aggregation value $v_{N,o,s}$, the algorithm checks if the value is already in the RAM cache. If it is true, the value is fetched and shuffled to the beginning of the RAM cache. In this way, we favor the values which are frequently queried. When there is no result in the RAM cache, we try to find the requested value in the Aggregation Cache. In case of success, the value is added to the RAM cache. If the required value does not exist neither in RAM nor Aggregated Cache, its calculation must be recursively done on the basis of the child nodes of N. Clearly, we can use the already calculated values of the child nodes of N, provided that they exist in RAM or Aggregation Cache.

During the calculation of v, we calculate its cost c using a formula proposed at the beginning of the section. The cost determines the Aggregation Cache partition where the value should be stored (see Algorithm 1, line 2). If the cost is high, then we put the aggregate value v in the memory bucket of the higher partition number. When the Aggregation Cache is full, all aggregated values from one bucket of the lowest partition are removed. Please note that the calculation cost is not stored explicitly.

Algorithm 1. CacheInsertion()

Input: AggregatedValue v
Input: Cost c
1: $p' \longleftarrow LowestPartitionNumber()$;
2: $p \longleftarrow EstimatePartition(v, c)$;
3: **if** $AggregationCache\ is\ not\ full$ **then**
4: $AddToPartition(p, v)$;
5: **else**
6: **if** $p' < p$ **then**
7: $RemoveBucketFromPartition(p')$;
8: $AddToPartition(p, v)$;
9: **end if**
10: **end if**

2.3 Aggregation Cache

In this subsection, we describe the Aggregation Cache in more detail.

Determining Initial Partitions. When the system starts, the Aggregation Cache is empty. In this situation, the calculation cost of the aggregated value connected with the node N depends on the node's level and the number of its

children. Let f_{min} and f_{max} be the minimal and maximal number of the children of each node (except the leaf nodes) in the R-tree, respectively.

Let l be the level number of the node N. Clearly, the number of its children at $l-1$ varies between f_{min} and f_{max}, at $l-2$ varies between f_{min}^2 and f_{max}^2 and so on.

In general, the minimal and maximal number of the subnodes of N may be estimated as: $\sum_{i=1}^{l-1} f_{min}^i$ and $\sum_{i=1}^{l-1} f_{max}^i$, respectively. Hence, the minimal and maximal cost of the aggregated value v_n is between: $r * \sum_{i=1}^{l-1} f_{min}^i$ and $r * \sum_{i=1}^{l-1} f_{max}^i$, where r denotes a read cost.

To calculate the expected cost of v_N, we pick a constant p between f_{min} and f_{max}. Then, the expected calculation cost of v_N is $r * \sum_{i=1}^{l-1} p^i$.

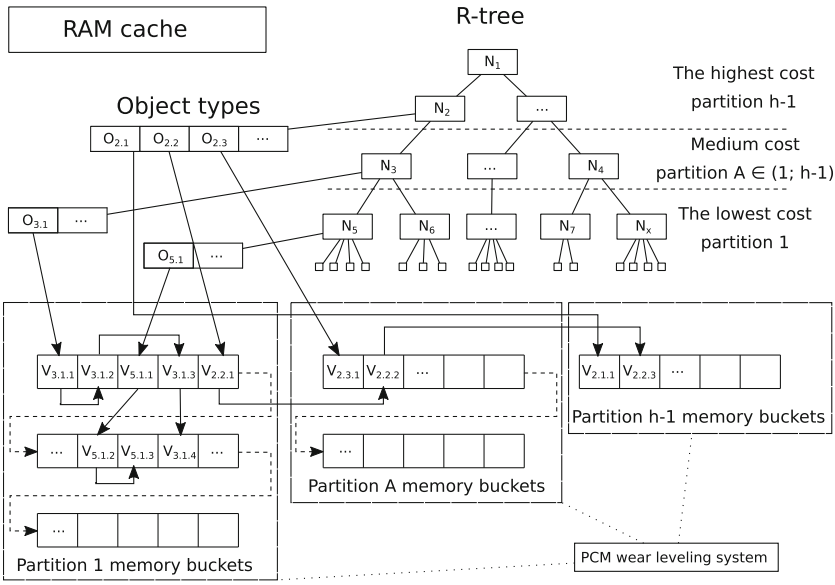


Fig. 3. ARM architecture

Now, we can create $h-1$ initial partitions. Please note that the leaf-level has no partition, because no calculation cost of the aggregated values is expected. Moreover, the root node belongs to the partition of the level below, because it is no point to create a partition for only one node.

Cost ranges for each partition are:

- Partition 1: 0 to $r * p$
- Partition l, for $0 < l < h-1$: $r * \sum_{i=1}^{l-1} p^i$ to $r * \sum_{i=1}^{l} p^i$
- Partition $h-1$: $r * \sum_{i=1}^{h-2} p^i$ to ∞

In our example (Fig. 3), the R-tree has 4 levels (without leaves). So, we create 3 partitions. Each partition consists of linked memory buckets. Please note

that the partitions correspond to the levels only when there are no aggregated values cached. The situation could be different when the aggregated values of all root's children are cached. In this case, the aggregated value of the root can be calculated quickly from cache without traversing the tree. As a consequence, its calculation cost would be small and the value would go to the first partition.

Implementation Issues. Because a number of child nodes of the R-tree varies between f_{min} and f_{max}, we cannot guess p which would generate partitions precisely matching calculation costs at each level. We use $p = average(f_{min}, f_{max})$ to reflect a common case when the R-tree has good node spread.

Total memory used by Aggregation Cache can be controlled by limiting maximal number of buckets generated by all partitions. After reaching the limit we delete the oldest bucket in the lowest partition.

Figure 3 is an example of our solution. The R-tree consists of four levels. A node may have a pointer to the object type array. It denotes that the node has the aggregated values of the object types stored in the Aggregation Cache. For example, the node N_2 has the aggregated values $v_{2,2,1}$ and $v_{2,2,2}$ of two different aggregation types associated with an object type $o_{2,2}$. Values assigned to one node can be put in different partitions, if their calculation costs are drastically different.

Let us suppose that the value $v_{3,1,1}$ connected with the node N_3 is requested but not stored in the Aggregation Cache. In this situation, $v_{3,1,1}$ must be calculated on the basis on its child nodes, i.e. N_5 and N_6. Since the value $v_{5,1,1}$ of N_5 is already stored, it can be directly fetched. However, the value of N_6 must be estimated recursively. As a result, the calculation cost of $v_{3,1,1}$ is much smaller than the cost in the situation when no aggregated values are cached. It is a reason why $v_{3,1,1}$ is not stored in the middle partition, but in the lowest one.

When R-tree or aggregated values of leaf nodes are updated, aggregate values of all upper nodes might end up invalidated. But changing the number of cars in an area does not affect the number of buildings. Only aggregates related to modified object type are invalidated. Invalid nodes can be found by traversing the R-tree. It helps us to reduce the number of writes by updating invalid aggregates without modifying the rest.

3 Performance Evaluation

In this section, we shall discuss some simulations which confirm the effectiveness of our method. In the simulations, we use byte-addressable PCM with read and write latency set to 50 ns and 1 µs, respectively. We created the R-tree, which consists of 5 levels and contains 100000 leaf nodes. Every single inner node has between 10 and 20 child nodes. We claim that the R-tree with such parameters is typical in the real applications. In the implementation, the Aggregation Cache has a fixed size. The size denotes the maximal number of aggregated values. Every experiment was written in Java 8.0 under Windows 7 and was conducted

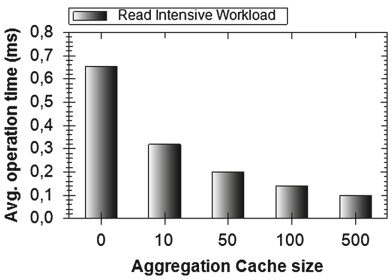


Fig. 4. Performance depending on the Aggregation Cache size for the read intensive workload

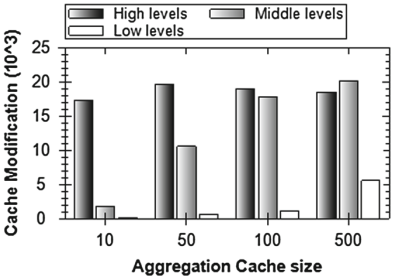


Fig. 5. Modification number for different workloads depending on the Aggregation Cache size

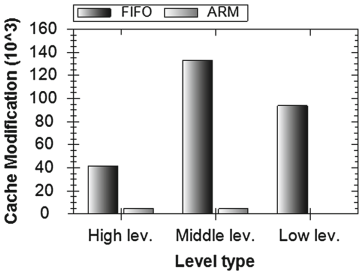


Fig. 6. FIFO vs. ARM depending on the query pattern

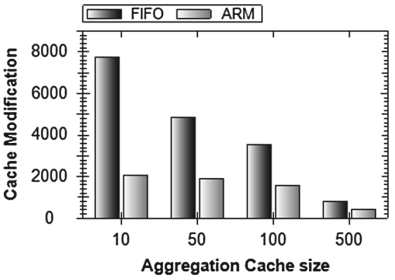


Fig. 7. FIFO vs. ARM depending on the Aggregation Cache size

on the machine equipped with the processor Intel Core(TM) i-5 3380M 2.90 GHz and 4 GB RAM.

In the first experiment, we present the average operation time depending on the Aggregation Cache size (Fig. 4). The Aggregation Cache size (axis X) denotes how many aggregated values can be stored in it (value 0 means that the cache is not used). We performed 100000 read operations and 10000 write operations. As we can see, the Aggregation Cache size affects the performance. When the size is small, the number of stored aggregated values is small as well and the query is executed slower.

In the second experiment (Fig. 5), we count Aggregation Cache modifications after requests of the nodes in different R-tree levels. It turns out that the modifications occur more often for the nodes in the highest levels of the R-tree (near the root node). The requests for the nodes at the bottom of the R-tree (levels: 1 and 2) do not change the data in the Aggregation Cache very frequently.

In the next experiments, we compare our approach with FIFO (First In First Out). We measure the number of Aggregation Cache modifications. As we mentioned previously, this is the most important aspect of PCM. Figures 6 and 7 show the difference between these two methods depending on the asked levels and Aggregation Cache size, respectively. Please note that when the low levels are

queried, the number of writes is minimal (Fig. 6) in the case of ARM. From the experiments we can conclude that our approach drastically outperforms FIFO. As it can be observed on Fig. 7, our method requires 4 to 2 times less write operations than FIFO. All other cache replacement policies described in [10] are more write intensive than FIFO. Their application would cause higher write traffic and could wear out PCM cells quicker.

We conducted much more evaluations which are not contained in this paper. We measured performance of the method for the write dominant workload in which a write operation occurs 10 times more frequently than a read one. In this case, the performance of both methods is similar. The experiments performed on the more extensive R-trees (i.e. with 500000 or 1000000 leaves) show very similar results to these presented in this section.

4 Conclusions and Future Work

In this paper, we propose the architecture of Aggregation R-tree Method (ARM) which stores the aggregated values on phase change memory. Our algorithm utilizes the PCM Aggregation Cache as a space where the aggregated values are stored. The method chooses aggregated values which storage is the most beneficial with regard to the computational performance. Out approach decreases the number of PCM writes and consequently outperforms the other methods (for example FIFO) on PCM.

For the future work, it would be interesting to find a method which could adjust the parameter p depending on the real query pattern.

Acknowledgement. The paper is supported by Wroclaw University of Science and Technology (grant number 0401/0017/17).

References

1. Qureshi, M.K., Karidis, J., Franceschini, M., Srinivasan, V., Lastras, L., Abali, B.: Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 42, pp. 14–23. ACM, New York (2009). doi.acm.org/10.1145/1669112.1669117
2. Ou, Y., Chen, L., Xu, J., Härder, T.: Wear-aware algorithms for PCM-Based database buffer pools. In: Chen, Y., et al. (eds.) WAIM 2014. LNCS, vol. 8597, pp. 165–176. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11538-2_16
3. Wu, Z., Jin, P., Yue, L.: Efficient space management and wear leveling for PCM-based storage systems. In: Wang, G., Zomaya, A., Perez, G.M., Li, K. (eds.) ICA3PP 2015. LNCS, vol. 9531, pp. 784–798. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27140-8_54
4. Chen, S., Gibbons, P.B., Nath, S.: Rethinking database algorithms for phase change memory (2011)
5. Viglas, S.D.: Write-limited sorts and joins for persistent memory. Proc. VLDB Endow. **7**(5), 413–424 (2014). https://doi.org/10.14778/2732269.2732277

6. Oh, G., Kim, S., Lee, S.W., Moon, B.: SQLite optimization with phase change memory for mobile applications. Proc. VLDB Endow. **8**(12), 1454–1465 (2015). https://doi.org/10.14778/2824032.2824044
7. Jabarov, E., On, B., Choi, G.S., Park, M.: R-tree for phase change memory. Comput. Sci. Inf. Syst. **14**(2), 347–367 (2017). http://doiserbia.nb.rs/Article.aspx?id=1820-02141700008J
8. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: International Conference on Management of Data, pp. 47–57. ACM (1984)
9. Pawlik, M., Macyna, W.: Implementation of the aggregated R-tree over flash memory. In: Yu, H., Yu, G., Hsu, W., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012. LNCS, vol. 7240, pp. 65–72. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29023-7_7
10. Swain, D., Dash, B.N., Shamkuwar, D.O., Swain, D.: Analysis and predictability of page replacement techniques towards optimized performance. In: IJCA Proceedings on International Conference on Recent Trends in Information Technology and Computer Science, pp. 12–16 (2012). Full text available

Modeling Query Energy Costs in Analytical Database Systems with Processor Speed Scaling

Boming Luo[1(✉)], Yuto Hayamizu[1], Kazuo Goda[1], and Masaru Kitsuregawa[1,2]

[1] The University of Tokyo, Tokyo, Japan
{luo,haya,kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp
[2] National Institute of Informatics, Tokyo, Japan

Abstract. Energy efficiency in analytical database systems is becoming increasingly important because of the rapid growth in energy consumed by data centers driven by the recent big data boom. Previous studies showed that processor speed scaling has the potential to improve energy efficiency of analytical queries. These results, however, were obtained from measurement of specific queries. The power–performance characteristics of processor speed scaling specific to analytical database systems still remains unexplored despite their importance in energy efficient analytical query processing. We tackle this problem by modeling the energy costs of analytical queries with processor speed scaling based on query processing throughput. Our experimental evaluation shows that our energy model can be fitted within an error of 1.65% and can be used to identify power–performance characteristics of analytical queries.

1 Introduction

Energy management is a primary concern in current data centers. The consumption of energy in data centers has been rapidly growing and is forecasted to reach 8% of worldwide energy production by 2020 [1]. Because the rate of worldwide data generation is increasing rapidly [2], an increasing amount of IT resources, e.g., servers and storage systems, have been installed in data centers to develop large-scale data analytics platforms. Therefore, energy efficiency in analytical database systems—the key component of the platforms—is becoming a serious concern.

Processor speed scaling, also referred to as Dynamic Voltage and Frequency Scaling (DVFS), is a well-known power–performance tuning knob. Prior studies of processor speed scaling adopted a *measurement-based approach* for investigating its power–performance characteristics in analytical database systems. Tsirogiannis et al. [3] analyzed power–performance profiles among various hardware configurations and reported that the higher operating frequency resulted in the better energy efficiency. In contrast, Götz et al. [4] demonstrated that the most energy-efficient operating frequency was not always the highest one and

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 310–317, 2018.
https://doi.org/10.1007/978-3-319-98812-2_27

largely varied depending on workload characteristics. While these measurement-based studies provided practical insights about the power–performance characteristics of processor speed scaling in analytical database systems and the guidelines on energy management specific to the measured queries, they cannot be applied to a wide spectrum of analytical queries.

Herein, we tackle a *model-based approach* to investigating the power–performance characteristics of processor speed scaling for analytical query processing. In this paper, we focus on modeling an energy cost of analytical queries comprising sequential scans as basic building blocks, which has not been studied in the literature as far as we know. The presented model enables us to quantitatively analyze the effect of processor speed scaling on power–performance characteristics of a wide range spectrum of analytical queries.

2 Analytical Database Systems with Processor Speed Scaling

Processor Speed Scaling. For improving energy efficiency in analytical database systems, processor speed scaling, also referred to as DVFS, is a well-known tuning knob for runtime energy management adaptive to workload shifts. Usually, a processor defines available frequency levels and preset voltage levels for each frequency level like Intel SpeedStep Technology in Intel Xeon processors [5], thus an operating system or an application program can manage the processor power consumption of processor cores through this interface. Some studies have reported that processor speed scaling can potentially improve the energy efficiency of analytical query processing [3,4,6,7]; however the power–performance characteristics of processor speed scaling in analytical databases are still largely unexplored mainly because their measurement-based approaches on specific queries.

Energy Saving Opportunity for Analytical Database System. Because energy efficiency is defined as query processing throughput per power consumption, the key problem is correctly quantifying the balance between these two metrics. When a query is compute intensive and query processing throughput is restricted by processor performance, query processing throughput and processor power consumption cased by query processing are approximately proportional to the operating frequency of the processor. Because power is also consumed by other components and peripherals, energy efficiency is considered higher for higher operating frequencies. Conversely, when query processing throughput is restricted by other factors, e.g., storage I/O, there are potential ways for reducing power consumption with little impact on performance. As sequential I/O patterns are known to account for the bulk of analytical query I/O workloads [8], instruction execution in the processor can be easily overlapped with I/O operations using common I/O read-ahead techniques. In this scenario, as long as processor computing throughput is higher than I/O throughput, lowering the operating frequency should not result in performance penalty and may possibly

improve energy efficiency. To the best of our knowledge, these opportunities for energy efficiency improvement in analytical database system have mentioned [9] in previous studies but have not been quantitatively modeled.

3 Throughput-Based Energy Cost Formulation

We take a model-based approach to identify the power–performance characteristics of processor speed scaling on analytical database server comprising processors, storage devices, memory modules, and other peripherals, e.g., a motherboard, cooling fans, and power supply unit. By modeling the power–performance characteristics, we can identify the behavior of power consumption and performance over operating frequencies and voltages of a processor. We focus on single query execution and analytical queries comprising sequential scans as basic building blocks as a first step toward understanding the power–performance characteristics of a vast variety of analytical workloads.

We adopt the pipeline model for energy cost modeling as it is the general concept in database systems and used in the literature [9,10]. For query execution, database system generates a query execution plan comprising a tree structure whose nodes are database operators. By grouping database operators which can be concurrently executed as a single pipeline, one or more subtrees are formed and we refer to these subtrees as *basic execution blocks*. The workload is steady and the fluctuate of the system power consumption is relatively small in a single basic execution block, thus we formulate an energy cost model at this granularity.

For basic execution blocks comprising sequential scan and join operators, the throughput θ (tuples per unit time) of a basic execution block can be limited either by the computational throughput θ^{CPU} of the processor, which is generally proportional to the operating frequency f of the processor, or by the I/O throughput θ^{IO} of the storage. Hence, θ can be expressed as follows:

$$\theta = \min\left(\theta^{\mathrm{CPU}}, \theta^{\mathrm{IO}}\right), \quad \theta^{\mathrm{CPU}} = af$$

where a is a coefficient that depends on the characteristics of the given processor and workload. Given that N is the number of tuples processed in a basic execution block, the execution time T of the basic execution block can be calculated as follows:

$$T = N/\theta = N/\min\left(af, \theta^{\mathrm{IO}}\right) \tag{1}$$

θ is limited by θ^{CPU} and T is inversely proportional to f when $f < \theta_{\mathrm{IO}}/a$, whereas T is independent of f when $f > \theta_{\mathrm{IO}}/a$. $f = \theta_{\mathrm{IO}}/a$ is the balance point between processor and storage throughput and we refer to this frequency as the *boundary frequency*.

Let us now consider power consumption of the system. We assume that power consumption of the active processor cores and storage devices stays steady within a single basic execution block and power consumption of other components of the

server stays constant regardless of the types of basic execution blocks [1]. Fan et al. [11] suggested a nonlinear model of processor power consumption of analytical workload as function of CPU utilization, therefore, we adopt this model. In this model, the difference in power consumption from the idle state ΔP^{CPU} can be expressed as $\Delta P^{\mathrm{CPU}} \propto 2u - u^r$ $(1 \leq r)$. On the assumption that u is solely determined based on θ, then u can be expressed as $u = \theta/\theta^{\mathrm{CPU}}$. It is known that the higher the operating frequency f, the higher the operating voltage V required for stable operation [12]. Although the actual relationship between f and V depends on processor implementations, we assume that $V(f)$ is a stepwise function of f following common implementations. Consequently, P^{CPU} can be expressed as follows:

$$P^{\mathrm{CPU}} = \left\{AfV^2 + Bf + C(V - V_{\mathrm{idle}})\right\}(2u - u^r) + P^{\mathrm{CPU}}_{\mathrm{idle}} \tag{2}$$

where A, B, C are coefficients and $V_{\mathrm{idle}}, P^{\mathrm{CPU}}_{\mathrm{idle}}$ represent the voltage and power dissipation during the idle state, respectively. In the curly brackets, each term corresponds to transition power dissipation, short-circuit power dissipation and leakage power dissipation.

Next, we assume that storage power consumption P^{IO} is proportional to the utilization of disk. P^{IO} can be expressed as follows:

$$P^{\mathrm{IO}} = D\theta/\theta^{\mathrm{IO}} + P^{\mathrm{IO}}_{\mathrm{idle}}$$

where D is the amount of increase in power under the maximum utilization of disk and $P^{\mathrm{IO}}_{\mathrm{idle}}$ is power dissipation during the idle state.

Finally, we assume that power consumption of other components (P^{others}) is constant. Thus, the total power consumption of system P is calculated as follows:

$$P = P^{\mathrm{CPU}} + P^{\mathrm{IO}} + P^{\mathrm{others}}$$

Because energy consumption E is the product of power consumption P and execution time T, E can be expressed as follows:

$$\begin{aligned} E &= PT = \left(P^{\mathrm{CPU}} + P^{\mathrm{IO}} + P^{\mathrm{others}}\right) T \\ &= N\left[\frac{AfV^2 + Bf + C(V - V_{\mathrm{idle}})}{\theta}(2u - u^r) + \frac{D}{\theta^{\mathrm{IO}}} + \frac{P^{\mathrm{CPU}}_{\mathrm{idle}} + P^{\mathrm{IO}}_{\mathrm{idle}} + P^{\mathrm{others}}}{\theta}\right] \end{aligned}$$

Let us now consider the operating frequency f_{opt} that minimizes energy consumption. The value of f_{opt} depends on where throughput-determining factor is. We assume that the processor has n levels of operating frequencies $(\{f_1, \cdots, f_n\}, f_i < f_{i+1})$ and there are m rising edges of $V(f)$ $(\{f_{s_1}, \cdots, f_{s_m}\})$ such that $V(f_{s_j}) > V(f_{s_j - 1})$. When throughput θ is limited by storage $(f >$

[1] Although this assumption does not always hold on realistic environments, modeling power consumption as a time-varying function and considering variation of power consumption of other components are beyond the scope of this paper.

θ_{IO}/a), E monotonically increases with f. When throughput θ is limited by processor ($f < \theta^{\mathrm{IO}}/a$), if there exists no rising edges in $V(f)$, E monotonically decreases with f. If not, E increases by the increase of $V(f)$ at a rising edge f_{s_j} and might take a local minimum value at f_{s_j-1}. Therefore, f_{opt} can be categorized into three cases and determined through the proposed model:

(a) the minimum frequency f_1, when $f_i > \theta^{\mathrm{IO}}/a$ ($i = \{1, \cdots, n\}$)
(b) the boundary frequency f_b or f_{s_j-1} ($s_j < b$), when there exists f_b that satisfies $f_b > \theta^{\mathrm{IO}}/a$ ($1 < b \leq n$) and $f_k < \theta^{\mathrm{IO}}/a$ ($k = \{1, \cdots, b-1\}$)
(c) the maximum frequency f_n or f_{s_j-1}, when $f_i < \theta^{\mathrm{IO}}/a$ ($i = \{1, \cdots, n\}$)

4 Experimental Evaluation

Experimental Environment. We used HP Z440 Workstation as the server, equipped with Intel Xeon E5-1603 v4, 8 GB memory, Seagate BarraCuda (3.5 in., 2 TB, 7200 rpm). The operating frequency ranged from 1.2 to 1.9 GHz and from 2.1 to 2.8 GHz in 0.1 GHz steps ($f = \{f_1, \cdots, f_{16}\}$). Power consumed by the whole server system was recorded at 20 Hz sampling using Yokogawa WT1800. We used Linux 3.10.0 as kernel, PostgreSQL 9.6.3 as DBMS and the TPC-H dataset with a scale factor of 100. We observed the system had a jump in power consumption at 2.8 GHz in micro-benchmark as preliminary experiment, thus the CPU operating voltage in our system was assumed to be V_1 when $f \leq 2.7$ GHz and V_2 when $f = 2.8$ GHz.

Query. Using the TPC-H dataset, we defined evaluation queries based on full table scan. These queries are named as queries A, B and C.

- **Query A**: $\sigma(\texttt{LINEITEM})$ with five aggregate expressions and two grouping variables.
- **Query B**: $\sigma(\texttt{LINEITEM})$ with an aggregate expression.
- **Query C**: $\sigma(\sigma(\texttt{ORDERS}) \bowtie \texttt{LINEITEM})$

We confirmed that full table scan was used for the execution plan of queries A and B. Additionally, full table scan and hash join were used for query C. The execution plans of both query A and B consisted of one basic execution block. The execution plan of query C consisted of two basic execution blocks because it performed hash join and we refer to them as C(1) and C(2).

Parameter Calibration. We measured the execution time and power consumption of queries A, B, and C for each operating frequency and then calibrated the parameters of each basic execution block in following steps: (i) obtained values of N for each basic execution block from the estimation of PostgreSQL query planner, (ii) calibrated the parameters which are not specific to basic execution blocks, and (iii) calibrated basic-execution-block-specific parameters.

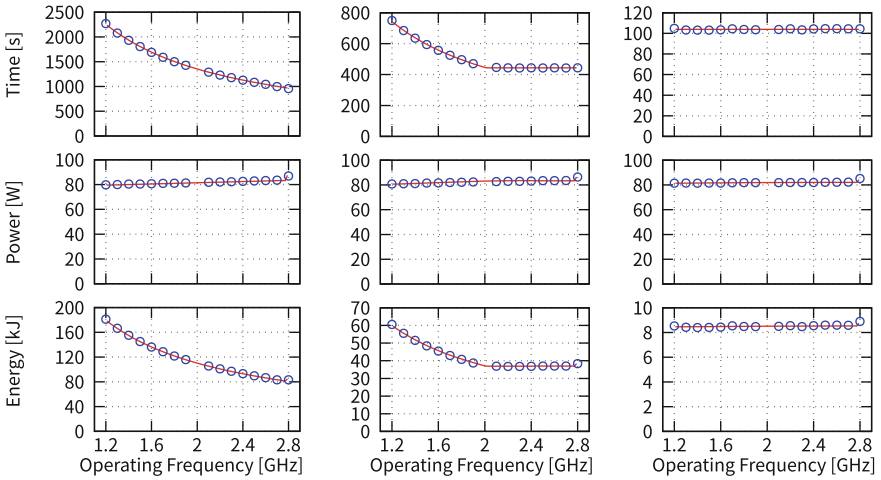


Fig. 1. Power–performance characteristics of processor speed scaling: measured (with blue circle) and modeled (with red line) (Color figure online)

Result. Figure 1 shows power-performance curves of each basic execution block (the result of query C(2) has omitted by the limitation of space). We can find the slope of the power consumption graph differed among the range of frequencies. In particular, the slope was smaller in the IO-bound condition in comparison with the CPU-bound condition. CPU utilization (u) was inversely proportional to operating frequency (f) when operation was IO-bound; hence, the effect of the increase in power consumption because of the increase in the operating frequency in Eq. (2) was nullified.

With regard to the frequency f_{opt} that minimized energy consumption, it was estimated to be 2.7 GHz for query A, while the actual minimum energy point was 2.8 GHz. The minimum energy point of queries B were 2.1 GHz by the model, whereas they were 2.2 GHz in the actual measurement. One possible cause of these discrepancies was a factor influencing execution time which the proposed model was unable to capture; however this requires further investigation. The increase in energy consumption because of these misestimation was 0.1% for query A, 0.4% for query B; however that for query C was 1.2%.

Overall, the measured results and the graph described by the fitted model almost agreed with each other in terms of execution time, power consumption, and energy consumption. The error in energy consumption between the measured value and the model-fitted value was within 1.65%. These results confirmed the effectiveness of the presented model.

5 Related Work

Processor speed scaling, also referred to as DVFS, is a common method for adjusting power and performance by switching the operating frequency and voltage of the processor. The basic approach of DVFS energy-saving policies is lowering the operating frequency at low CPU utilization to reduce power consumption with small performance penalty, as the general method taken by OS. In the context of transaction processing workload, there are studies about application-aware control of processor speed scaling by constructing power model [13,14] or SLA-based control mechanism [14,15].

For analytical query processing, the energy efficiency of hardware configurations have been comprehensively investigated in the late 2000s [16,17]. Recently, there have been several studies on energy-aware query optimization techniques and frameworks [18–20]. With regard to energy management with processor speed scaling, conventional approaches were measurement-based. Tsirogiannis et al. [3] extensively measured the energy efficiency of processor configurations with various types of analytical queries, analyzed the power–performance profiles, and concluded that the highest frequency tended to be the most energy efficient. Contrary to their study, Götz et al. [4] reported that the best frequency varied based on query workloads and proposed a technique for calibrating frequency with workload recording and query benchmarking. Manousakis et al. [7] proposed a feedback DVFS controller using real-time power sensor measurement values. In contrast to these studies, we adopted a model-based approach to formulate the power–performance characteristics of analytical query processing with processor speed scaling.

6 Conclusion

In this paper, we took a model-based approach to identifying power–performance characteristics of processor speed scaling for analytical query processing. We presented throughput-based formulation of an energy cost model. The presented model enabled us to quantitatively analyze the effectiveness of processor speed scaling for analytical queries comprising sequential scans as basic building blocks. The experimental evaluation showed that the energy model agreed with observations within an error of 1.65%.

As part of our future research, we plan to expand our model to workloads with index table scan or sorting and to examine the effectiveness of the proposed model on various hardware configurations.

References

1. Chalise, S., et al.: Data center energy systems: current technology and future direction. In: PES GM 2015, July 2015
2. Reinsel, D., Gantz, J., Rydning, J.: Data age 2025: the evolution of data to life-critical. Issue paper, April 2017

3. Tsirogiannis, D., Harizopoulos, S., Shah, M.A.: Analyzing the energy efficiency of a database server. In: SIGMOD 2010, pp. 231–242, June 2010
4. Götz, S., et al.: Energy-efficient databases using sweet spot frequencies. In: UCC 2014, pp. 871–876, February 2014
5. Intel: Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor. White paper, March 2004
6. Lang, W., Patel, J.: Towards eco-friendly database management systems. In: CIDR 2009 (2009)
7. Manousakis, I., Marazakis, M., Bilas, A.: FDIO: a feedback driven controller for minimizing energy in I/O-intensive applications. In: HotStorage 2013, June 2013
8. Yu, P.S., Chen, M.S., Heiss, H.U., Lee, S.: On workload characterization of relational database environments. IEEE Trans. Softw. Eng. **18**(4), 347–355 (1992)
9. Roukh, A., Bellatreche, L.: Eco-processing of OLAP complex queries. In: Madria, S., Hara, T. (eds.) DaWaK 2015. LNCS, vol. 9263, pp. 229–242. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22729-0_18
10. Kunjir, M., Birwa, P.K., Haritsa, J.R.: Peak power plays in database engines. In: EDBT 2012, pp. 444–455, March 2012
11. Fan, X., Weber, W.D., Barroso, L.A.: Power provisioning for a warehouse-sized computer. In: ISCA 2007, June 2007
12. Flynn, M.J., Hung, P.: Microprocessor design issues: thoughts on the road ahead. IEEE Micro **25**(3), 16–31 (2005)
13. Korkmaz, M., Karsten, M., Salem, K.: Towards dynamic green-sizing for database servers. In: Proceeding of ADMS 2015 (2015)
14. Xu, Z., Wang, X., Tu, Y.C.: Power-aware throughput control for database management systems. In: Proceeding of ICAC 2013, pp. 315–324 (2013)
15. Hayamizu, Y., Goda, K., Nakano, M., Kitsuregawa, M.: Application-aware power saving for online transaction processing using dynamic voltage and frequency scaling in a multicore environment. In: ARCS 2011, pp. 50–61, February 2011
16. Meza, J., Shah, M.A., Ranganathan, P., Fitzner, M., Veazey, J.: Tracking the power in an enterprise decision support system. In: ISLPED 2009, pp. 261–266, August 2009
17. Poess, M., Nambiar, R.O.: Tuning servers, storage and database for energy efficient data warehouses. In: ICDE 2010, pp. 1006–1017, March 2010
18. Xu, Z., Tu, Y.C., Wang, X.: PET: reducing database energy cost via query optimization. VLDB **5**(12), 1954–1957 (2012)
19. Roukh, A., Bellatreche, L., Ordonez, C.: EnerQuery: energy-aware query processing. In: CIKM 2016, pp. 2465–2468, October 2016
20. Tu, Y.C., Wang, X., Zeng, B., Xu, Z.: A system for energy-efficient data management. ACM SIGMOD Rec. **43**(1), 21–26 (2014)

Graph Querying and Databases

Sprouter: Dynamic Graph Processing over Data Streams at Scale

Tariq Abughofa and Farhana Zulkernine(✉)

Queen's University, Kingston, ON K7L 2N8, Canada
{abughofa,farhana}@cs.queensu.ca

Abstract. Graph data is becoming dominant for many applications such as social networks, targeted advertising, and web indexing. As a result of that, advances in machine learning and data mining techniques depend tightly on the ability to process this data structure efficiently and reliably. Despite the importance of processing dynamic graphs in real-time, it remains a challenge to maintain such graphs and process them over data streams. We propose Sprouter, an end-to-end framework which enable storing enormous graph data, allows updates in real-time, and supports efficient complex analytics in addition to OLTP queries. We demonstrate that our framework can ingest and process streaming data efficiently using a scalable multi-cluster distributed architecture, apply incremental graph updates, and store the dynamic graph for fast query performance. Experiments showed the system is able to update graphs having up to 100 million edges in under 50 s in a moderate underlying cluster. As we use all open source tools, the framework can be easily extended in the future with other equivalent software.

Keywords: Graph processing · Stream data processing
Big data management and analytics

1 Introduction

Graph processing is one of the most important topics in Big Data processing. The graph structure is suitable for distributed processing as processing works in an iterative manner allowing parallelism. Also, the structure has proved to be suitable in representing social networks, targeted advertising, and web page indexes. However, many of the data applications nowadays require processing data in real-time. That should be reflected in graph processing as well. Streams of data are processed as they pass through streaming data processing engines and as a result the graph should be updated in real-time. We refer to this problem as *dynamic graph processing* or *incremental graph processing* [2,8].

Although graph-specific processing systems, such as Pregel [6] and GraphJet [7], enable high performance, they have shortcomings. Analytic applications generally require multiple steps of data processing such as map-reduce jobs, SQL-like queries, and machine learning algorithms. Such systems are not sufficient for

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 321–328, 2018.
https://doi.org/10.1007/978-3-319-98812-2_28

implementing these pipelines. These problems led to developing *general-purpose data processing systems with graph abstractions* on top of them.

Apache Spark [9] was the leader in this direction as it introduced an embedded graph framework called *GraphX* [4] and later GraphFrames [3]. Since Spark is an iterative processing system by nature, GraphX was developed as a graph abstraction library that sufficiently expresses graph APIs using basic dataflow operators such as reduce, join, and map. A graph in GraphX and GraphFrames is represented as a pair of vertex and edge collections stored as distributed partitions and is highly fault tolerant like RDDs and DataFrames.

Although Spark supports stream processing using the Spark Streaming library [10], evolving the graph over data streams is not presented as a feature in GraphX. Abughofa et al. [1] explored finding a solution for incremental graph processing on top of Spark with efficient pinpoint lookups and updates. The paper looked into using three solutions: RDDs- the Spark memory abstraction, IndexedRDDs- an in-memory key-value store designed for efficient fine-grained updates, and Redis- an in memory datastore. The paper demonstrated that IndexedRDDs have limitations in sequential updates/lookups and that Redis, although performs good, is a complicated solution for property graph. RDDs, on the other hand, are not designed to perform well with pinpoint queries/updates.

In this paper, we propose Sprouter[1], a framework that can process real-time data streams using Apache Spark, and thereby, update data in an associated very large dynamic in-memory graph structure. We illustrate the functionality of the framework by ingesting streams of social data and keeping it up to date with new relations coming with the stream. This will enable, for example, giving recommendations about whom one should interact with or follow to overcome the cold-start problem. We show that a graph with up to 100 million edges can be updated in under 50 s in a moderate underlying cluster. The framework supports online queries and complex graph analytics such as PageRank.

The rest of the paper is organized as follows. In Sect. 2, we talk about the related work. In Sect. 3, we present an overview of the proposed framework. The experiments we conducted to test incremental graph updates and pin-point queries are presented in Sect. 4 along with the results and a critical discussion. Finally, we conclude and provide guidelines for future work in Sect. 5.

2 Related Work

Iyer et al. [5] presented GraphTau, a graph processing framework built on top of GraphX to process dynamic graphs. However, the framework is not open-sourced and is built on the RADIX trees similar to IndexedRDDs, which do not serve pinpoint lookups on graph streams as mentioned earlier.

Choudhury et al. [2] designed a framework with Spark to process dynamic knowledge graphs. The authors implemented many applications. However, none of the implemented applications require evolving the graph in real-time. Pattern

[1] https://github.com/TariqAbughofa/sprouter.

discovery is done over data streams but it uses the triplet stream passed to Spark to find patterns and thus incrementing the graph in real-time is not needed. Another application, question answering, is done with graph search but over a static graph which is loaded from HDFS and no streaming is done. Our focus is to find a solution that enables real-time incremental graph processing.

3 The Sprouter Framework

The framework requires the components shown in Fig. 1. The design decisions and the setup are explained next.

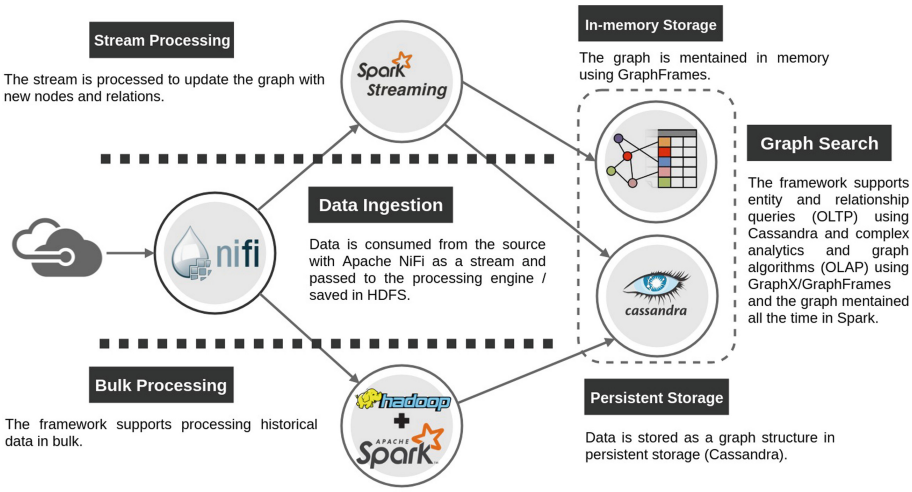


Fig. 1. The Sprouter framework design

3.1 Data Ingestion

We chose using Apache NiFi[2] which supports building data flows through a web-based drag-and-drop graphical interface without the need to write any code.

This component perform two tasks: First, it ingests the historical data and stores it in HDFS for bulk processing. Second, it ingests the most recent data as a stream every 15 min for real-time processing.

3.2 Graph Bulk Processing and Stream Processing

We need to support bulk processing to able to process historical data as well as data left in the queue in case of downtime. We chose Apache Spark as our processing engine for many reasons. First, it is a unified engine that supports

[2] https://nifi.apache.org/.

bulk and stream processing which simplifies the framework design. Second, it has two graph abstractions that allow efficient graph processing namely GraphX and GraphFrames. Third, it is a reliable mature engine that has proven efficiency and scalability [11].

3.3 Persistent and In-memory Storage

We keep the graph in-memory for efficient graph analytics with Spark. Evolving the graph with GraphFrames, however, is not a straightforward operation. Listing 1.1 shows how to increment the graph using GraphFrames in Scala.

Listing 1.1. Graph Appending in GraphFrames

```
val fullVertices = oldGraph.vertices
  .join(newVertices, Seq("name"), "outer")
  .select("name").withColumn("id", monotonically_increasing_id)
val fullEdges = graph.edges.union(newEdges)
val fullGraph = GraphFrame(fullVertices, fullEdges).cache()
```

RDDs are not efficient for pinpoint lookups especially with fine-grained updates over streaming data as found by Abughofa et al. [1]. The same paper stated that no out of the box solution exists today to address this problem. To support simple graph queries such as vertex lookup or getting edges connected to a certain vertex, we need to use an external graph-based storage system.

The storage system has to be a scalable distributed NoSQL datastore that is write-efficient and performs well for Big Data. The system should have the ability to save/load Spark Streaming micro-batches efficiently in bulks. It also needs to support OLTP queries such as finding the neighbors of a vertex. In addition to that, finding high availability with eventual consistency is definitely preferred over strict consistency.

Our initial candidates were HBase and Cassandra. Both databases are scalable NoSQL systems that proved to have good performance for Big Data applications. However, HBase will not have bulk loading functionality until its third release[3]. In addition to that it does not offer full availability in order to ensure full consistency. Cassandra, on the other hand, has bulk loading/saving and highly available.

We chose to adapt the HGraphDB[4] storage model on top of Cassandra instead of other models such as JanusGraph to achieve greater scalability and efficiency for broader scope of queries. HGraphDB uses separate tables for the edges and the vertices.

3.4 Graph Search

We are concerned with two types of graph search: simple OLTP queries and OLAP or analytical queries such as PageRank and Community Detection. GraphX and GraphFrames are not efficient for OLTP queries as explained earlier

[3] https://issues.apache.org/jira/browse/HBASE-14150.
[4] https://github.com/rayokota/hgraphdb.

but Cassandra provides an efficient solution for that. However, GraphFrames is designed to provide highly flexible API with many pre-implemented graph algorithms as well as the ability to build custom ones. That why we decided that any OLAP or analytical queries would be executed on the in-memory graph maintained all the time in GraphFrames.

4 Experiments

4.1 Environment and Experimental Details

We used 6 identical machines each with 8 cores 2.10 GHz Intel Xeon CPU, 64-bit architecture, 24 GB of RAM, and 300 GB of disk space. We installed the Hortonworks Data Platform (HDP-2.6.3.0) on our machines with Spark v 2.2.0. Cassandra (v 3.11.1) is installed on all 6 nodes to form a cluster. Two of the six Cassandra servers are used as seed nodes for the cluster, which serve as contact points for the new nodes that attempt to join a cluster.

For all the experiments, we used 12 executors with 3 cores each and 5 GB of RAM. We validated our framework by demonstrating dynamic and distributed graph updates from streaming data as new data is appended to an existing graph. We also executed pin-point queries on the graphs in both Cassandra and GraphFrames.

Listing 1.2. A record from the test data in JSON format excluding irrelevant fields

```
{
  "GKGRECORDID": "20180128150000-0",
  "V2.1DATE": "20180128150000",
  "V2SOURCECOMMONNAME": "msn.com",
  "V2DOCUMENTIDENTIFIER": "https://www.msn.com/en-ca/news/canada/no-one-denied-flight-
       because-of-no-fly-list-mistakes-government-memo-says/ar-BBIlaAr", ...
  "V2ENHANCEDPERSONS": [
    { "name": "Scott Bardsley", "offset": "1082" },
    { "name": "Justin Trudeau", "offset": "1690" },
    { "name": "Catharine Tunney", "offset": "199" },
    { "name": "Khadija Cajee", "offset": "2045" } ...
  ], ...
}
```

We used The Global Data on Events, Location, and Tone (GDELT)[5] as a source for our data. GDELT contains data from news media from all over the world in print, broadcast, and web formats. In this study, we used the Global Knowledge Graph (GKG) v2.1 which contains the full news graph connecting persons, organizations, locations, emotions, events, and news sources. We demonstrate a record of the data in Listing 1.2.

The file is parsed and the persons/offset list is extracted and then sorted by offset. We define that two people have a connection if they appear in the same paragraph and that the connections are undirected. For simplicity, we assumed that it means that they have at most 1000 character distance (the average length of a paragraph). We then drop self-relations and duplicates.

[5] https://www.gdeltproject.org/.

4.2 Results

To conduct the experiments, we consumed historical data from the GDELT repository for the years of 2017 and 2018. This data allowed us to construct a graph consisting of 13,290,365 vertices and 97,134,816 edges using Spark, which we stored in Cassandra in the HGraphDB format as we described earlier. Then we ran the streaming job which loads the graph using Spark Streaming and reads the GDELT "last" file. We made sure the stream consumes always the same file for all the experiments to ensure result consistency. The file we chose contained information about 12,738 articles which generated 51,060 vertices and 87,099 edges that needed to be appended to the full graph.

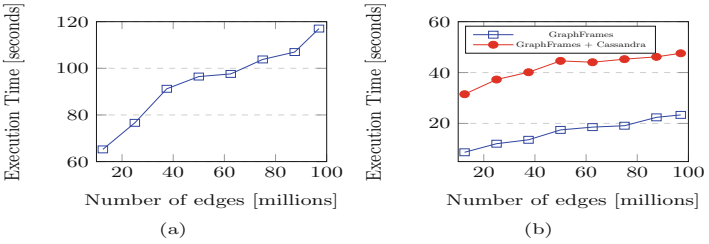


Fig. 2. Appending data: (a) directly to GraphFrames (b) into Cassandra first then loading it to GraphFrames (Color figure online)

We implemented two approaches for the append operation. **The first approach** we described earlier in Listing 1.1. The new data is merged with the GraphFrames graph directly. In Fig. 2a, we show the time it took to update different sizes of graphs in GraphFrames. The plot in blue shows the time spent to append the graph in GraphFrames without persisting in Cassandra. The red plot shows the total time to update the in-memory graph and store the updated graph in Cassandra. In practice we will need to persist new data in Cassandra, and therefore, the red plot represents the actual time to append a graph in the framework. Figure 2a that the time increases linearly with the graph size from 8.66 s for a graph with 125 m edges to 23.35 s for a graph with 97 m edges.

Figure 2b shows the time for **the second approach**. In this solution we first update the tables in Cassandra with the new edges and vertices. Then we load the complete graph from Cassandra to GraphFrames as a bulk loading operation as described before. This approach showed bad execution time that increases with graph size which is expected since it reads the full data from disk to memory. The slope for this approach is bigger and the execution time starts at 65.23 s for a graph of size 12.5 m edges and increases to 117 s for a graph having 97 m edges. This also shows that appending in GraphFrames directly remains a better solution as the graph grows.

Table 1 gives a summary of the performances of appending graphs of various sizes by updating GraphFrames only (GF), both GraphFrames and Cassandra

Table 1. Summary of the experiment numerical results

Graph	A	B	C	D	E	F	G	H
Links	12.5 m	25 m	37.5 m	50 m	62.5 m	75 m	87.5 m	97 m
Nodes	4.7 m	7 m	8.4 m	9.5 m	10 m	11 m	11.9 m	13 m
GF	8.66 s	12.03 s	13.56 s	17.45 s	18.55 s	19.1 s	22.39 s	23.35 s
$GF \rightarrow C$	31.48 s	37.26 s	40.12 s	44.59 s	44.05 s	45.26 s	46.17 s	47.57 s
$C \rightarrow GF$	65.23 s	76.6 s	91.2 s	96.46 s	97.59 s	103.8 s	106.97 s	117.03 s

($GF \rightarrow C$), and updating Cassandra first then loading to GraphFrames ($C \rightarrow GF$). The table displays computation times in seconds for each approach/graph.

In addition to having good performance, the first approach can be adapted to apply periodic updates to Cassandra on each GraphFrame update since GraphFrames are inherently fault-tolerant. This can greatly decrease the operational cost and processing time which we intend to explore further as a future work.

To prove that Cassandra is needed for OLTP graph queries, we chose two simple queries: Vertex lookup by Id and vertex neighbors lookup by querying the edge table/RDD. We used *Graph H* from Table 1. Results (Fig. 3) show big advantage of using Cassandra for such queries. We also executed PageRank on GraphFrames as an example of OLAP queries. Such algorithms cannot be executed on Cassandra.

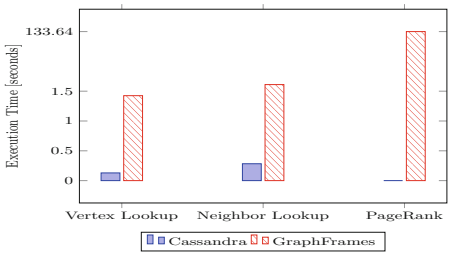


Fig. 3. Graph queries: Cassandra vs GraphFrames

5 Conclusion

Graph analytics applications where data is streamed continuously require dynamic graph processing in real-time. In this paper, we proposed Sprouter, a framework to process and store dynamic graphs over data streams using Apache Spark. The framework design has many contributions. In contrast to most graph processing systems which are built to process static graphs, the graph construction is viewed as an incremental process as the graph evolves with time. The system supports executing simple queries as well as complex algorithms on the dynamically updated graph. Also all components scale to accommodate huge growth of data.

The experiments showed that the framework can append graphs up to 100 million edges in under 50 s and allows efficient pin-point queries. Our framework can be applied to complex streaming data analytics applications like image processing and community detection using data from multiple domains such as text and IoT. The in-memory graph enables more complex multi-level data analytics and knowledge linking for decision support.

In the future, we plan to explore other big data stores and graph databases. We are working on determining the optimal interval for periodic update of disk storage and testing a community detection algorithm for dynamic graphs.

References

1. Abughofa, T., Zulkernine, F.: Towards online graph processing with spark streaming. In: IEEE International Conference on Big Data, pp. 2787–2794. IEEE (2017)
2. Choudhury, S., et al.: NOUS: Construction and querying of dynamic knowledge graphs. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 1563–1565. IEEE (2017)
3. Dave, A., Jindal, A., Li, L.E., Xin, R., Gonzalez, J., Zaharia, M.: Graphframes: an integrated API for mixing graph and relational queries. In: Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, p. 2. ACM (2016)
4. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graphx: Graph processing in a distributed dataflow framework. In: OSDI, vol. 14, pp. 599–613 (2014)
5. Iyer, A.P., Li, L.E., Das, T., Stoica, I.: Time-evolving graph processing at scale. In: Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, p. 5. ACM (2016)
6. Malewicz, G., et al.: Pregel: a system for large-scale graph processing. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 135–146. ACM (2010)
7. Sharma, A., Jiang, J., Bommannavar, P., Larson, B., Lin, J.: Graphjet: real-time content recommendations at twitter. Proc. VLDB Endow. **9**(13), 1281–1292 (2016)
8. Yin, S., et al.: Node-grained incremental community detection for streaming networks. In: IEEE 28th International Conference on Tools with Artificial Intelligence, pp. 585–592. IEEE (2016)
9. Zaharia, M., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, p. 2. USENIX Association (2012)
10. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I.: Discretized streams: fault-tolerant streaming computation at scale. In: Proceedings of the 24th ACM Symposium on Operating Systems Principles, pp. 423–438. ACM (2013)
11. Zaharia, M., et al.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016)

A Hybrid Approach of Subgraph Isomorphism and Graph Simulation for Graph Pattern Matching

Kazunori Sugawara(✉) and Nobutaka Suzuki

University of Tsukuba, 1-2 Kasuga, Tsukuba, Ibaraki 305-8550, Japan
s1721674@s.tsukuba.ac.jp, nsuzuki@slis.tsukuba.ac.jp

Abstract. There are two major approaches for graph pattern matching: subgraph isomorphism and graph simulation. With the former approach, each answer is a subgraph that exactly matches a given pattern graph. However, this feature is sometimes too restrictive since strongly-related nodes may be in different answers even if such nodes should be in the same answer. With the latter approach, we can find an answer efficiently but the answer is a single relation in which a number of unrelated subgraphs are intermingled together. In this paper, we propose a hybrid approach of subgraph isomorphism and graph simulation for graph pattern matching. The distinctive feature of our approach is that we can find answer subgraphs so that each answer subgraph consists of strongly-related subgraphs that match a given pattern and that unrelated subgraphs are not intermingled in the same answer unlike graph simulation.

1 Introduction

Graph pattern matching is one of the most important problem for querying graphs. There are two major approaches for graph pattern matching: subgraph isomorphism and graph simulation. For a pattern graph P and a data graph G, the former problem is to find all the answers (subgraphs) in G that exactly match P. However, this feature is sometimes too restrictive since strongly-related nodes may be broken up into different answers even if such nodes should be in the same answer. Moreover, subgraph isomorphism is NP-complete and solving the problem requires exponential computation cost. On the other hand, the latter problem, graph simulation, is to find a single relation of nodes in G that can match nodes of P, which can be obtained efficiently. However, a number of unrelated subgraphs are intermingled in the same relation together.

In this paper, we propose a hybrid approach of subgraph isomorphism and graph simulation for graph pattern matching. In our approach, a pattern graph has two kinds of nodes: key nodes and the others, where the key nodes can be arbitrarily chosen by users. Here, key nodes are the nodes to which the notion of subgraph isomorphism is applied, while the non-key nodes are the nodes to which the notion of graph simulation is applied. Let $P = (V_p, E_p)$ be a pattern graph

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 329–338, 2018.
https://doi.org/10.1007/978-3-319-98812-2_29

and $K \subseteq V_p$ be the set of key nodes. Since K can be chosen arbitrarily, our approach covers arbitrary intermediate points between subgraph isomorphism and graph simulation. By using our approach, we can find answers so that each answer consists of strongly-related nodes that match a given pattern and that unrelated nodes are not intermingled in the same answer unlike graph simulation.

As a simple example, let us consider the pattern graph P and the data graph G shown in Fig. 1. In Fig. 1, u_1 represents a student, u_2 represents a professor, and u_3 represents a course. Figure 2 lists the answers obtained by (a) subgraph isomorphism, (b) graph simulation, and (c) our approach (assuming that u_1 is the key node of the pattern). Note that by using u_1 as the key node, in our approach an answer is obtained for *each* student node. In this case, answers are "grouped" by s_1 and s_2. Thus, courses c_2 and c_3 taken by s_1 are grouped into the same answer in Fig. 2(c). Such an answer cannot directly be obtained by subgraph isomorphism; c_2 and c_3 are in distinct answers as shown in Fig. 2(a).

Based on the hybrid approach, we propose an algorithm that computes answers from a given pattern graph and a data graph. Alternatively, an answer of our approach could be obtained by aggregating the relevant answers of subgraph isomorphism, e.g., the first answer of Fig. 2(c) is obtained by aggregating the first and the second answers of in Fig. 2(b). However, such a naive approach is clearly inefficient. Therefore, we devise an algorithm for computing answers "directly" from a pattern graph and a data graph, which is more efficient than the naive approach. Experimental results show that our algorithm can find answers efficiently, especially for "complex" graphs containing various kinds of nodes.

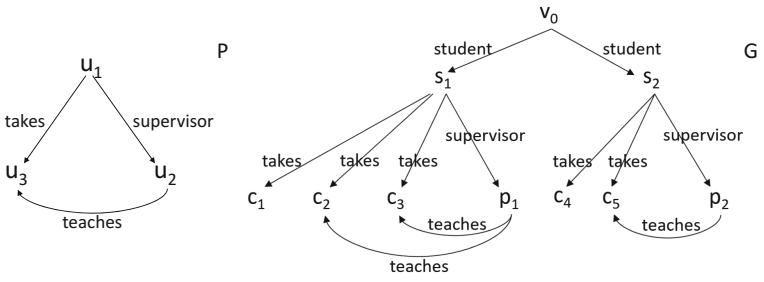


Fig. 1. Example of pattern and data graph

Related Work

A number of algorithms for solving subgraph isomorphism are proposed. The algorithm proposed by Ullmann is the first practical algorithm for this problem [9]. Since the problem is NP-complete, aiming at more efficient computation, a number of heuristic algorithms, e.g., VF2 [6], QuickSI [8], and CFL-Match [1], are proposed. The common feature of these algorithms is *backtracking* [9]. That is, the current partial answer is extended by a node, and if the extended partial answer is invalid, then we backtrack to the previous partial answer and the other "next" node is examined. Our approach can be applied to these backtracking algorithms for solving subgraph isomorphism. For graph simulation, a

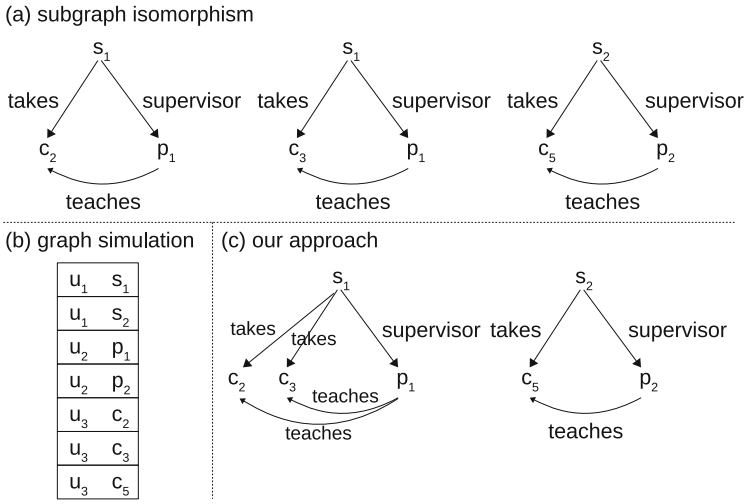


Fig. 2. Answers of three approaches

polynomial-time algorithm for solving this problem is proposed by Henzinger [3]. Moreover, several extensions to graph simulation are proposed, e.g., [2,5]. To the best of our knowledge, no study has been made to hybridize the notion of subgraph isomorphism and graph simulation.

2 Preliminary Definitions

A *labeled directed graph* (*graph*, for short) is denoted by $G = (V, E)$, where V is a set of nodes and E is a set of labeled directed edges. By $e = u \xrightarrow{l} v$ we mean an edge e from a node u to a node v labeled by l. For a node $v \in V$, the set of incoming edges of v is denoted $In(v)$ and the set of outgoing edges of v is denoted $Out(v)$. By $lab_{in}(v)$ we mean the set of labels of incoming edges of v. That is, $lab_{in}(v) = \{l \mid u \xrightarrow{l} v \in In(v), u \in V\}$. Similarly, by $lab_{out}(v)$ we mean the set of labels of outgoing edges of v. That is, $lab_{out}(v) = \{l \mid v \xrightarrow{l} u \in Out(v), u \in V\}$.

Both a pattern (query) and data to be queried are defined as graphs. Let $P = (V_p, E_p)$ be a pattern graph and $G = (V, E)$ be a data graph. Then subgraph isomorphism and graph simulation are formulated as follows.

Subgraph isomorphism

Find all injective functions $f : V_p \to V$ such that for any $u \xrightarrow{l} u' \in E_p$, there exists an edge $f(u) \xrightarrow{l} f(u') \in E$.

Grpah (dual) simulation

Find a binary relation $S \subseteq V_p \times V$ such that for any $u \xrightarrow{l} u' \in E_p$ the following conditions hold.

- If $(u, v) \in S$, then there exists an edge $v \xrightarrow{l} v' \in E$ such that $(u', v') \in S$.
- If $(u', v') \in S$, then there exists an edge $v \xrightarrow{l} v' \in E$ such that $(u, v) \in S$.

3 Our Approach

3.1 Hybridizing Subgraph Isomorphism and Graph Simulation

We extend the notion of pattern graph by "keys". Formally, an *extended pattern graph* (*pattern graph*, for short) is defined as a triple $P = (V_p, K, E_p)$, where $((V_p \cup K), E_p)$ is a graph, $V_p \cap K = \emptyset$, and each node in K is called *a key*. In short, the notion of subgraph isomorphism is applied to the nodes in K and the notion of graph simulation is applied to the nodes in V_p. Let $f : K \to V$ be an injective function and $S \subseteq V_p \times V$ be a binary relation. For a pattern graph $P = (V_p, K, E_p)$ and a data graph $G = (V, E)$, we say that a pair (f, S) is an *answer* of P on G if f and S satisfy the following conditions.

- For each edge $u \xrightarrow{l} u' \in E_p$, the following conditions hold.
  - The case where $u \in K$: there exists an edge $v \xrightarrow{l} v' \in E$ such that $f(u) = v$ and that $f(u') = v'$ if $u' \in K$, $(u', v') \in S$ otherwise.
  - The case where $u \in V_p$: for each $(u, v) \in S$ there exists an edge $v \xrightarrow{l} v' \in E$ such that $f(u') = v'$ if $u' \in K$, $(u', v') \in S$ otherwise.
- For each edge $u' \xrightarrow{l} u \in E_p$, the following conditions hold.
  - The case where $u \in K$: there exists an edge $v' \xrightarrow{l} v \in E$ such that $f(u) = v$ and that $f(u') = v'$ if $u' \in K$, $(u', v') \in S$ otherwise.
  - The case where $u \in V_p$: for each $(u, v) \in S$ there exists an edge $v' \xrightarrow{l} v \in E$ such that $f(u') = v'$ if $u' \in K$, $(u', v') \in S$ otherwise.

We say that an answer (f, S) of P on G is *maximum* w.r.t. f if for any answer (f, S') of P on G, $S' \subseteq S$.

3.2 Algorithm

We present an algorithm for finding answers of a pattern graph and a data graph. Although our hybridization approach does not depend on a specific subgraph isomorphism algorithm, to describe our algorithm specifically we need to pick up a specific subgraph isomorphism algorithm and hybridize our approach and the algorithm. Therefore, the overall description of our algorithm follows the QucickSI [8] description in [4], since QuickSI is simple to describe and gives a good performance among the algorithms compared in [4].

Visit Order of Pattern Graph. Let P be a pattern graph and G be a data graph. Our algorithm HybridMatch uses a minimum spanning tree of P in order to determine the visit order of the nodes in P (this part is similar to QuickSI). We assign a weight to each edge in P based on the frequency of edge labels in G, and HybridMatch constructs a minimum spanning tree of P by using Prim's algorithm. We use table SEQ to record the minimum spanning tree. In SEQ, $T_i.node$ denotes a node, $T_i.parent$ denotes the parent of $T_i.node$, $T_i.dir$ denotes the direction of the edge between $T_i.parent$ and $T_i.node$, and $T_i.l$ denotes the

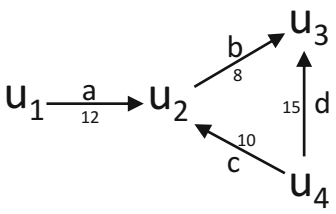


Fig. 3. Weighted pattern graph

Table 1. *SEQ* for the graph in Fig. 3

Type	$T_i.node$	$[T_i.dir, T_i.l, T_i.parent]$
T_0	u_1	
T_1	u_2	[in, a, u_1]
T_2	u_3	[in, b, u_2]
T_3	u_4	[out, c, u_2]
R_1	u_4	[out, d, u_3]

label of the edge. Extra rows R_j may occur in SEQ to record *extra edges*, i.e., edges that does not appear in the minimum spanning tree. For example, Fig. 3 shows a weighted pattern graph P and Table 1 shows SEQ of P. As indicated in SEQ, the algorithm visits u_1, u_2, u_3, u_4 in this order.

Algorithm 1. HybridMatch

Input: weighted pattern graph $P = (V_p, K, E_p)$, data graph $G = (V, E)$
Output: all answers (f, S) of P on G
1: Initialize SEQ to an empty table
2: $V_T := \emptyset$
3: Choose a node $u \in K$, add u to V_T, and add u to SEQ
4: **while** $|V_T| \neq |V_p| + |K|$ **do**
5: $P := \{e \mid e \in E_p$ such that $e.u \in V_T \wedge e.u' \notin V_T\}$
6: Select an edge $e \in P$ such that $w(e)$ is minimum in P
7: Add e to SEQ, $V_T := V_T \cup \{e.u'\}$
8: **for** each $e \in E_p \backslash SEQ$ such that $e.u \in K \wedge e.u' \in K$ **do**
9: Add e to SEQ as an extra edge
10: Let f be an empty function
11: $S := \emptyset$
12: $d := 0$
13: Search(P, G, SEQ, f, S, d)

Details of the Algorithm. We present the details of HybridMatch. Let $P = (V_p, K, E_p)$ be a pattern graph and $G = (V, E)$ be a data graph. First, HybridMatch picks up a node $u \in K$ and add u to V_T and SEQ (line 3). Next, starting from u, HybridMatch finds a minimum spanning tree in P based on edge weight $w(e)$, in which the edges are added to SEQ in the order chosen in the spanning tree (lines 4 to 7). Then HybridMatch adds the rest of edges to SEQ as extra edges (lines 8 and 9). Finally, HybridMatch invokes a recursive procedure Search to find all answers (f, S) of P on G (line 13).

We next describe Search (Algorithm 2). In Search, we use a notion of *dangling* nodes. In short, dangling nodes are redundant ones that violate the simulation and should be deleted from S. Formally, if for some edge $u' \xrightarrow{l} u \in E_p, v \in S(u)$ but E contains no edge $v' \xrightarrow{l} v$ such that $v' \in S(u')$, then we say that v is

dangling. Similarly, if for some edge $u' \xrightarrow{l} u \in E_p, v' \in S(u')$ but E contains no edge $v' \xrightarrow{l} v$ such that $v \in S(u)$, then we say that v' is *dangling*.

Algorithm 2. Search(P, G, SEQ, f, S, d)

if $d = |V_p| + |K|$ **then**
  for each extra edge $u \xrightarrow{l} u' \in SEQ$ **do**
    if $f(u) \xrightarrow{l} f(u') \notin E$ **then**
      return
  $S := S \backslash \{(u, v) \in S \mid f(u') = v$ for every $u' \in K\}$
  DeleteDangling(P, G, f, S)
  if $S(u) \neq \emptyset$ for every $u \in V_p$ **then**
    report (f, S)
else
  $u := T_d.node \in SEQ$
  $C(u) :=$ FilterCandidates(P, G, SEQ, f, S, d)
  if $u \in K$ **then**
    for each $v \in C(u)$ such that v is not yet matched **do**
      $f(u) := v$
      Search$(P, G, SEQ, f, S, d + 1)$
      $f(u) := nil$
  else
    $S := S \cup \{(u, v) \mid v \in C(u)\}$
    Search$(P, G, SEQ, f, S, d + 1)$
    $S := S \backslash \{(u, v) \mid v \in C(u)\}$

Algorithm 3. FilterCandidates(P, G, SEQ, f, S, d)

Input: weighted pattern graph $P = (V_p, K, E_p)$, data graph $G = (V, E)$, injective function $f : K \to V$, relation $S \subseteq V_p \times V$, depth d
Output: a set of candidate nodes $C(u)$

$T := T_d \in SEQ$
$u := T.node$, $u_{par} := T.parent$
if $d = 0$ **then**
  $C(u) := \{v \in V \mid lab_{in}(u) \subseteq lab_{in}(v) \wedge lab_{out}(u) \subseteq lab_{out}(v)\}$
  return $C(u)$
if $u_{par} \in K$ **then**
  if $u \xrightarrow{l} u_{par} \in E_p$ **then**
    $C(u) := \{v \mid v \xrightarrow{l} f(u_{par}) \in E\}$
  else if $u \xleftarrow{l} u_{par} \in E_p$ **then**
    $C(u) := \{v \mid v \xleftarrow{l} f(u_{par}) \in E\}$
else
  if $u \xrightarrow{l} u_{par} \in E_p$ **then**
    $C(u) := \{v \mid v \xrightarrow{l} v' \in E \wedge (u_{par}, v') \in S\}$
  else if $u \xleftarrow{l} u_{par} \in E_p$ **then**
    $C(u) := \{v \mid v \xleftarrow{l} v' \in E \wedge (u_{par}, v') \in S\}$
return $C(u)$

Algorithm 4. DeleteDangling(P, G, f, S)

```
1: Q := ∅
2: for each u ∈ V_p do
3:    if S(u) contains a dangling node then
4:       S_old(u) := S(u)
5:       Delete every dangling node from S(u), and add u to Q
6: while Q ≠ ∅ do
7:    Delete a node u' from Q
8:    for each v' ∈ S_old(u')\S(u') and each v' -l-> v ∈ E such that u' -l-> u ∈ E_p ∧
      (u, v) ∈ S do
9:       if v becomes dangling then
10:         S_old(u) := S(u)
11:         Delete v from S(u) and add u to Q
12:   for each v' ∈ S_old(u')\S(u') and each v -l-> v' ∈ E such that u -l-> u' ∈ E_p ∧
      (u, v) ∈ S do
13:      if v becomes dangling then
14:         S_old(u) := S(u)
15:         Delete v from S(u) and add u to Q
```

Search finds nodes matched with P by recursively traversing G. d denotes the depth of recursive call. When $d = |V_p|+|K|$, i.e. Search reaches the bottom of the recursion, the algorithm does the following (lines 1 to 8). First, Search checks if each extra edge in SEQ has its corresponding edge in G, and returns no answer if some extra edge has no its corresponding edge (lines 2 to 4). Then Search deletes nodes included in f (i.e., key nodes) from S, and also deletes dangling nodes from S by invoking DeleteDangling (lines 5 and 6). Finally, Search reports a pair (f, S) as an answer (line 8). If $d < |V_p|+|K|$, Search first invokes FilterCandidates (Algorithm 3) to find a set $C(u)$ of candidate nodes which matches a dth node u in SEQ. Once $C(u)$ is obtained, we go back to Search. If the dth node u is a *key*, then for each candidate nodes $v \in C(u)$, Search updates f (line 14), recursively call Search (line 15), and restore the status at line 14 (line 16). Otherwise, Search updates S (line 18), recursively call Search (line 19) and restore the status at line 18 (line 20).

3.3 Correctness and Time Complexity of the Algorithm

We briefly show the correctness of the algorithm, and then consider the time complexity of the algorithm. Proofs are omitted because of space limitation.

Theorem 1. *Let R be the result of* HybridMatch *for pattern graph $P = (V_p, K, E_p)$ and data graph $G = (V, E)$. Then $(f, S) \in R$ iff (f, S) is a maximum answer of P and G.*

Let us next consider the time complexity of HybridMatch. We have the following theorem.

Theorem 2. *For a pattern graph $P = (V_p, K, E_p)$ and a data graph $G = (V, E)$,* HybridMatch *runs in $O(V_{PL}^{|K|+1} + |E| \cdot V_{PL}^2)$ time, where $V^{in}(l)$ is the set of nodes having an incoming edge labeled by l, $V_{PL}^{in} = \max_{l \in \Sigma_P} |V^{in}(l)|$, V_{PL}^{out} is similar to V_{PL}^{in}, and $V_{PL} = \max(V_{PL}^{in}, V_{PL}^{out})$. In particular, we have the following.*

1. *If $|K| \leq c$ for some constant c, then* HybridMatch *runs in polynomial time.*
2. *If $|K| \in O(\log |E_p|)$, then* HybridMatch *runs in quasi-polynomial time.*

Thus the complexity of the algorithm depends on the size of K. If K is "maximum", i.e., K coincides with the set of nodes of P, i.e., $V_p = \emptyset$, then our matching problem becomes equivalent to subgraph isomorphism which is NP-complete. However, the conditions of the above theorem implies that HybridMatch runs efficiently as long as the size of K is "modest" w.r.t. the size of P. In most cases, some small value suffices as the size of K. For example, if a user want to extract information of journals from a bibliographic database, then it suffices to set "journal" node as the key. Therefore, we believe that our approach works efficiently in most practical cases.

4 Experimental Evaluation

In this section, we present experimental results of our proposed algorithm. We implemented our algorithm in Ruby, and all the experiments were conducted on a machine with Intel Xeon E5-2623 v3 3.0 GHz CPU, 16 GB RAM, 2TB SATA HDD, and Linux CentOS 7 64bit. We used the following two datasets.

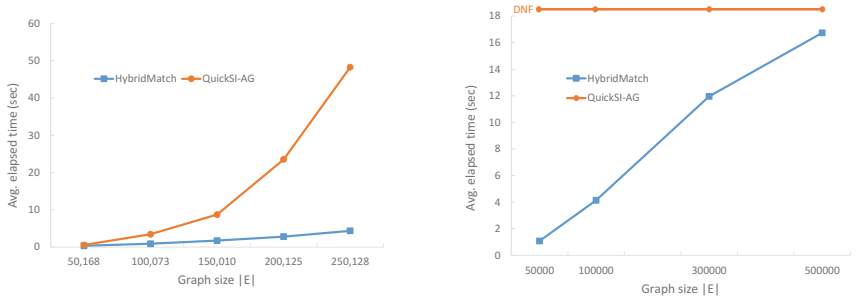
- SP2Bench [7] is a benchmark tool for generating RDF files based on DBLP. We generated five graphs with 50K, 100K, 150K, 200K, and 250K edges.
- DBPedia consists of structured information extracted from Wikipedia. We downloaded a file of graph[1] and created four graphs with 50K, 100K, 300K, and 500K edges by extracting the first n lines (i.e., n edges) of the file.

We measured the average elapsed time w.r.t. the size of data graph and the size of K. In this experiment, we made random pattern graph generators for the DBPedia and SP2Bench datasets. We generated 10 pattern graphs each with 4, 6, 8, and 10 edges, therefore we used 40 pattern graphs in total for each dataset.

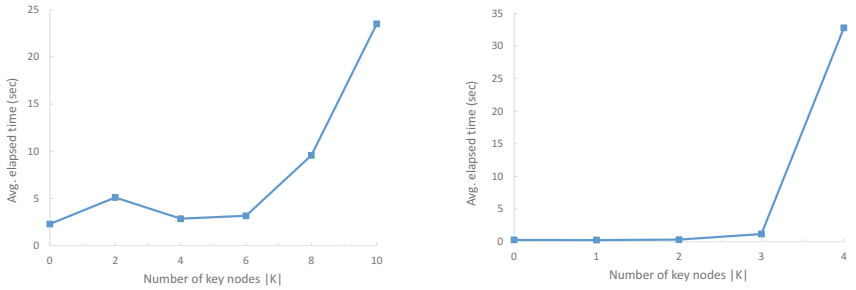
The answers of our approach can be obtained by running a subgraph isomorphism algorithm and then "aggregating" the obtained answers by key nodes. Therefore, we compare the elapsed times of such aggregating approach and our approach. In this experiment, the aggregating approach is implemented by the QuickSI algorithm [8] since our algorithm description is based on QuickSI and thus comparing our approach and the QuickSI-based aggregating approach is the most reasonable. In the following, the aggregating approach is treated as the baseline and referred to as QuickSI-AG.

We ran HybridMatch with $|K| = 3$ and QuickSI-AG for each pattern using both datasets. We set $|K| = 3$ since in many cases three or lower value suffices

[1] http://benchmark.dbpedia.org/.



(a) elapsed time for SP2Bench w.r.t. data size **(b)** elapsed time for DBPedia w.r.t. data size



(c) elapsed time for SP2Bench w.r.t. $|K|$ **(d)** elapsed time for DBPedia w.r.t. $|K|$

Fig. 4. Results of experiment

as the size of K. Figure 4a and b show the results, where x-axis represents the size $|E|$ of graph and "DNF" stands for "at least one Did Not Finished in 24 h". As shown in Fig. 4a, HybridMatch is consistently faster than QuickSI-AG for the SP2Bench dataset. For the DBPedia dataset, QuickSI-AG did not finish within 24 h for every cases. Therefore, for both datasets our approach is more effective to obtaining hybridized answers.

Secondly, to measure the elapsed time w.r.t. the size of K, we carried out the following two experiments.

- For the SP2Bench dataset, we fixed a graph with 250, 128 edges and ran HybridMatch varying $|K|$ from 0 to 10 in 2 steps.
- For the DBPedia dataset, we fixed a graph with 50,000 edges and ran HybridMatch varying $|K|$ from 0 to 4 (for this graph, HybridMatch did not finish within 24 h when $|K| \geq 5$).

Figure 4c and d show the results, where x-axis represents $|K|$ in a pattern graph $P = (V_p, K, E_p)$. The average elapsed time increases when $|K|$ increase on both dataset. This is largely consistent with the time complexity shown in previous section.

Comparing the two datasets, the SP2Bench graphs are much simpler and more regular than the DBPedia graphs. SP2Bench graphs follows DBLP and have highly "regular" data structures, while DBPedia graphs have more "complex" structures containing variety kinds of nodes. This could affect the elapsed time. For SP2Bench, HybridMatch finished within a few to 25 s, while did not finished within 24 h with $|K| \geq 5$ for DBPedia. Therefore, HybridMatch works efficiently for "complex" graphs as long as K is kept to be small, but we need to improve the efficiency of HybridMatch for such "complex" graphs.

Finally, let us briefly compare HybridMatch and QuickSI-AG. For the SP2Bench graph, QuickSI-AG took more than 50 s (as shown in Fig. 4a), while HybridMatch took at most 25 s (Fig. 4c). For the DBPedia graph, QuickSI-AG did not finish within 24 h, while HybridMatch finished within 35 s whenever $|K| \leq 4$. Thus, our approach is much more efficient than the "aggregating" approach.

5 Conclusion

We proposed a hybrid approach of subgraph isomorphism and graph simulation for graph pattern matching. In our approach, the nodes of a pattern graph are divided into key nodes and the others. Key nodes are the nodes to which the notion of subgraph isomorphism is applied, while the non-key nodes are the nodes to which the notion of graph simulation is applied. We also presented an algorithm for solving our problem. Experimental results showed that our approach is more efficient than the "aggregating" approach.

References

1. Bi, F., Chang, L., Lin, X., Qin, L., Zhang, W.: Efficient subgraph matching by postponing cartesian products. In: Proceedings of the 2016 International Conference on Management of Data. SIGMOD 2016, pp. 1199–1214 (2016)
2. Fan, W., Wang, X., Wu, Y.: Incremental graph pattern matching. ACM Trans. Database Syst. **38**(3), 18:1–18:47 (2013)
3. Henzinger, M.R., Henzinger, T.A., Kopke, P.W.: Computing simulations on finite and infinite graphs. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science. FOCS 1995, pp. 453–462 (1995)
4. Lee, J., Han, W.-S., Kasperovics, R., Lee, J.-H.: An in-depth comparison of subgraph isomorphism algorithms in graph databases. In: Proceedings of the 39th International Conference on Very Large Data Bases. PVLDB 2013, pp. 133–144 (2013)
5. Ma, S., Cao, Y., Fan, W., Huai, J., Wo, T.: Strong simulation: capturing topology in graph pattern matching. ACM Trans. Database Syst. **39**(1), 4:1–4:46 (2014)
6. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. Pattern Anal. Mach. Intell. **26**(10), 1367–1372 (2004)
7. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: a SPARQL performance benchmark. In: ICDE 2009, pp. 222–233 (2009)
8. Shang, H., Zhang, Y., Lin, X., Yu, J.X.: Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. Proc. VLDB Endow. **1**(1), 364–375 (2008)
9. Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23**(1), 31–42 (1976)

Time Complexity and Parallel Speedup of Relational Queries to Solve Graph Problems

Carlos Ordonez[1(✉)] and Predrag T. Tosic[2]

[1] University of Houston, Houston, USA
[2] Washington State University, Pullman, USA

Abstract. Nowadays parallel DBMSs compete with graph Hadoop Big Data systems to analyze large graphs. In this paper, we study the processing and optimization of relational queries to solve fundamental graph problems, giving a theoretical foundation on time complexity and parallel processing. Specifically, we consider reachability, shortest paths from a single vertex, weakly connected components, PageRank, transitive closure and all pairs shortest paths. We explain how graphs can be stored on a relational database and then we show relational queries can be used to efficiently analyze such graphs. We identify two complementary families of algorithms: iteration of matrix-vector multiplication and iteration of matrix-matrix multiplication. We show all problems can be solved with a unified algorithm with an iteration of matrix multiplications. We present intuitive theory results on cardinality estimation and time complexity considering graph size, shape and density. Finally, we characterize parallel computational complexity and speedup per iteration, focusing on joins and aggregations.

1 Introduction

Graph analytics remains one of the most computationally intensive tasks in big data analytics. This is due to large graph size (going beyond memory limits), graph structure (complex interconnectivity) and the potential existence of an exponential number of patterns (e.g. paths, cycles, cliques). On the other hand, graph problems are becoming more prevalent on every domain, there is a need to query graphs and more graph-structured data sets are now stored on SQL engines. There has also been recent interest in the Hadoop world revisiting recursive queries with SPARQL [19]. In our opinion, even though query optimization is a classical, well studied topic optimization of relational queries on graphs needs further research. We focus on the evaluation of relational queries which solve a broad class of graph problems including reachability, shortest paths, network flows, PageRank and connected components.

Previous research revisited relational query processing to analyze graphs on parallel DBMSs [4,12], showing a DBMS is faster than Hadoop "Big Data" systems like GraphX and showing a columnar DBMS is significantly faster than a

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 339–349, 2018.
https://doi.org/10.1007/978-3-319-98812-2_30

traditional row DBMS. These papers presented many experimental time performance comparisons and scalability analyses showing the impact of optimizations and efficiency of algorithms. In contrast, in this paper we focus on theoretical issues. To that end, we unify two broad classes of graph algorithms, we study relational query processing and we present important theory results characterizing time complexity and parallel speedup. Given the foundational focus of this paper and space limitations, we do not present experiments, but the reader can find extensive experimental evaluation in [4,12].

2 Definitions

2.1 Graphs from a Mathematical Perspective

Without loss of generality we consider directed graphs because undirected graphs can be represented including two directed edges per vertex pair. Let $G = (V, E)$ be a directed graph with $n = |V|$ vertices and $m = |E|$ edges. G can contain cycles (simple paths starting and ending at the same vertex) and cliques (hidden complete subgraphs). We will show such graph patterns make graph algorithms slower. From an algorithmic perspective, G is processed as an $n \times n$ adjacency matrix E where E contains binary or real entries (i.e. each edge is present or not, or it has a distance). Notice we overload E to make notation intuitive. Then algorithms use matrix E as input in various computations based on an iteration of matrix multiplication. In general, we assume G is a sparse graph, where $m = O(n)$. Some harder problems, not tackled in this paper, include the traveling salesman problem (TSP) and clique detection. Detecting cliques with k vertices when k is not constant is a computationally hard problem, under the usual assumption of computational complexity ($\mathbf{P} \neq \mathbf{NP}$), because there is an exponential search space for cliques.

2.2 Graph Problems Solved with Matrices and Vectors

We study two complementary classes of problems: (1) Iterating matrix-vector multiplication, where the solution is an n-vector S. For notational convenience S is a column vector manipulated as a matrix $n \times 1$. (2) Iterating matrix-matrix multiplication, where the result is an $n \times n$ matrix R.

These two classes account for most common problems in graph analytics to analyze social networks and the Internet. Some graph problems worth mentioning that do not fall into these categories are the traveling salesman problem (TSP, which requires visiting every vertex) and clique detection/enumeration, which requires exploring an even larger search space.

Iterating Matrix-Vector Multiplication

The core iteration is $S_1 = E^T \cdot S_0$, $S_2 = E^T \cdot S_1$, and so on, where S_0 has a different initialization depending on the graph algorithm. That is, each new solution is fed back into the same matrix-vector multiplication (somewhat similar to the

Gauss-Seidel iterative method to solve linear equations). We would like to point out that for notational convenience it is better to use matrix-vector multiplication instead of vector-matrix multiplication, which would require using many transposition operators in equations.

Representative problems include single-source reachability, single-source shortest path, weakly connected components and PageRank, among others. Single-source algorithms initialize the desired vertex i entry in S to 1 ($S[i] = 1$) and all other entries to zero ($S[j] = 0$ when $i \neq j$). Weakly connected components initialize $S[i] = i$ for $i = 1 \ldots n$. PageRank uses a transition matrix T with probabilities obtained from E, initialized as $T_{ij} = E_{ji}/outdegree(j)$, where outdegree$(j) = E^T \cdot e_j$ (e_j is the canonical vector with 1 at j and zero elsewhere arriving from any vertex to i). Intuitively, in single-source reachability and shortest path each iteration gets further vertices reachable from chosen vertex i, starting with vertices reachable with one edge. For weakly connected components each vertex in a component gets labeled with the minimum id of common neighbors. PageRank iterates a special multiplication form $S_{I+1} = T \cdot S_I$ until probabilities of random walks stabilize following a Markov Chain process [7]. Although not central to graph analytics, it is noteworthy Topological Sort can also be expressed with this iteration.

Iterating Matrix-Matrix Multiplication

The basic iteration is to multiply E by itself: $E \cdot E$, $E \cdot E \cdot E$, and so on, which can be expressed as $R_j = R_{j-1} \cdot E$, where $R_0 = E$. Intuitively, we generalize single source reachability to any vertex to a broader reachability from any vertex to any vertex. An important observation is that for directed graphs multiplication is not commutative, because $E \cdot E \neq E \cdot E^T$. Only for undirected graphs $E \cdot E = E \cdot E^T$ because E is symmetric. When E is binary we can treat each vector-vector multiplication as boolean vectors or as real vectors. For each boolean vector dimension pair we compute a logical AND, then we compute a logical OR across all of them. For real vectors the power matrix E^k (E multiplied by itself $k-1$ times) counts the number of paths of length k between each pair of vertices, and it is defined as: $E^k = \Pi_{i=1}^{k} E$ (i is a local variable here, not to be confused with vertex id i used in queries). Considering E a boolean matrix the transitive closure G^+ edge set is $\mathbf{E}^+ = E \vee E^2 \vee \cdots \vee E^n$. On the other hand, if E is treated as a real matrix, this iterative multiplication algorithm has these operator changes: scalar multiplication becomes scalar addition and the sum() aggregation becomes min() or max() aggregations (resulting in the shortest or longest path).

2.3 Graphs Stored and Processed in a Relational Database

To make exposition more intuitive, we prefer to use the term "table" instead of "relation", to make the connection with SQL explicit. G is stored in table E as a list of edges, which is an efficient storage mechanism for sparse graphs. Let E be stored as a table with schema $E(i, j, v)$ with primary key (i, j), where v represents a numeric value (e.g. distance, weight). If there is no edge between

two vertices or v is zero then the edge is not stored. Table E is the input for relational queries using columns i and j to compute joins, as explained below.

3 Query Processing to Solve Graph Problems

3.1 Iterating a Query Evaluating Matrix-Vector Multiplication

Solution vector S is stored on table $S(i, v)$, where v stores some value $v > 0$. Value v can be a binary flag, the shortest path length, vertex id, or page probability value, depending on the algorithm. From an algorithmic perspective all problems can be solved by a unified matrix-vector multiplication algorithm, under a semiring, exchanging mathematical operations. We use $|S_0|$ to denote the initial number of rows in table S. $|S_0|$ depends on the algorithm, which impacts query processing time. For reachability and single-source shortest path $|S_0| = 1$ (because only the ith entry is present in table S, with 1 and ∞ respectively), whereas for weakly connected components and PageRank $|S_0| = n$ (with vertex ids and probabilities of random walks respectively).

Then the iteration $S \leftarrow E^T \cdot S$ translates into relational queries as $S \leftarrow E \bowtie_{E.i=S.i} S$. For reachability and single-source shortest path $|S_0| = 1$, whereas for weakly connected components and PageRank $|S_0| = n$. Therefore, initialization divides time complexity into two complexity classes. Since the iteration has a linear right recursion the query plan is a right-deep tree with E on the left child, S_{I-1} on the right child and S_I as the parent node. In general, an aggregation is computed at each iteration resulting in this generalized query:

$$S_I \leftarrow \pi_{i:f(E.v \bullet S.v)}(E \bowtie_{E.i=S.i} S_{I-1}),$$

where $f()$ is an aggregation (sum(),min(),max()) and $\bullet$ is a scalar operation (addition +, multiplication *). Notice we use π as a general projection operator, capable of computing aggregations. Since $\bullet$ is commutative, associative and has an identity element (0 for +, 1 for *), and $f()$ is distributive over $\bullet$ both operators together acting on numbers (real, integer) represent a *semiring*. Such algebraic property allows solving all problems with the same template algorithm, by exchanging mathematical operators.

3.2 Iterating a Query Evaluating Matrix-Matrix Multiplication

We define R as a generalized recursive view which allows solving a broad class of problems. Let R be the result table returned by a linearly recursive query, with schema $R(d, i, j, p, v)$ and primary key (d, i, j), where d represents maximum number of edges in path, i and j identify an edge at a specific length (recursion depth), p and v are computed with aggregations. In this paper, p counts the number of paths and v is an aggregated numeric value (recursively computed, e.g. shortest distance). We assume a recursion depth threshold k is specified to make the problem tractable and to derive $O()$ bounds.

In general, the Seminaive algorithm is presented in the context of First Order Logic (FOL) and Datalog [1]. Here we revisit the Seminaïve algorithm [2,3], with relational queries, using as input E, defined in Sect. 2. At a high level, we initialize $R_0 \leftarrow E$ and the basic iteration of Seminaive is $R_I \leftarrow R_{I-1} \bowtie E$, which is a linear recursion on R_I. That is, it stops when $\Delta = R_I \bowtie E = \emptyset$. Then Seminaive produces a sequence of partial tables $R_1, R_2, \ldots, R_k$ and the result R is the incremental union of all partial results: $R = \bigcup_j R_j$. Seminaive stops when R reaches a fixpoint, meaning no more edges were added [1]. This means $R_I = \emptyset$ or R is a complete graph. The join condition and required projection to make tables union-compatible are explained below. We emphasize that we use the most efficient version of Seminaïve algorithm [1] because we assume linear recursion (i.e., we only need Δ to stop). Since the number of iterations and time complexity required by Seminaive heavily depend on G structure we bound recursion with k s.t. $k \ll n$ (assuming G is sparse and n is large).

We now explain relational queries in more technical detail. Because recursion is linear and we have a bound k it can be transformed into an iteration of $k-1$ joins. The initialization step produces $R_1 = E$ and the recursive steps produce $R_2 = R_1 \bowtie_{R_1.j=E.i} E = E \bowtie E$, $R_3 = R_2 \bowtie_{R_2.j=E.i} E = E \bowtie E \bowtie E$, and so on. The general form of a recursive join is $R_{d+1} = R_d \bowtie_{R_d.j=E.i} E$, where the join condition $R_d.j = E.i$ creates a new edge between a source vertex and a destination vertex when they are connected by an intermediate vertex. Notice that at each recursive step a projection (π) is required to make the k partial tables union-compatible. We use R_k to represent the output table obtained by running the full iteration of $k-1$ joins on E, where E appears k times. The final result table is the union of all partial results: $R = R_1 \cup R_2 \cup \cdots \cup R_k$. The query plan is a deep tree with $k-1$ levels, k leaves (E) and $k-1$ internal nodes ($\bowtie$).

$$R_{d+1} \leftarrow \pi_{d,i,j,f(p),g(v)}(R_d \bowtie_{R_d.j=E.i} E), \tag{1}$$

where $f()$ and $g()$ represent SQL aggregations (e.g. sum, count, max). To make notation from Eq. 1 more intuitive we do not show either π or the join condition between R and E: $R_{d+1} = R_d \bowtie E$.

Proposition. Transitive closure can be solved with matrix multiplication. Therefore, it can be solved with a unified algorithm. Let $G^+ = (V, E^+)$ be the transitive closure graph and let E have binary entries and $b(E) = F$ be a matrix function that transforms the E matrix to binary form: $F_{ij} = 1$ when $E_{ij} > 1$ and $F_{ij} = 0$ otherwise. Then $E^+ = b(E + E^2 + \cdots + E^k) = E \vee b(E^2) \cdots \vee b(E^k)$. The 2nd expression is evaluated more efficiently with bit operators: $\wedge$ (and) instead of *, $\vee$ (or) instead of + (i.e. like Warshall's algorithm [2]).

Proposition. If $|E| = 1$ or $|E| = n(n-1)$ then $G = G^+$.

Proof: If $|E| = 1$ then $E^2 = 0$ so $E^+ = E$. For the second case $E^2 = E$. Therefore, in both cases $E^+ = b(E + E) = E$.

This proposition touches two extreme cases and it is important because it exhibits the optimal cases for Seminaïve, when it stops after just one iteration.

3.3 Time Complexity

Matrix-Vector Multiplication

Time to join E with S for a sparse graph is $O(n \log(n))$, where $|E| = \Theta(n)$. By a similar reasoning, time to compute the aggregation is the same. On the other hand, time complexity can reach $O(n^2 \log(n))$ with a very dense graph (i.e. similar to a complete graph, or having large cliques). Therefore, time complexity for each matrix-vector multiplication is $O(n \log(n))$ for a sparse graph and $O(n^2 \log(n))$ for a very dense graph.

Matrix-Matrix Multiplication

We now analyze time complexity iterating matrix-matrix multiplication considering different graphs of different structure and connectivity.

We focus on analyzing space and time complexity for the most challenging case: the join in iterative matrix-matrix multiplication. By a similar reasoning, $O()$ to compute the sum() aggregation is the same. We start with space complexity. Cardinality estimation is one of the hardest and most well-known problems in query processing [6]. We explore time complexity and query evaluation cost based on G characteristics. The goal is to understand how shape, density and connectivity impact $O()$ and cardinality of each partial result table.

Our first goal is to understand $|R_2| = |E \bowtie E|$. That is, the result of the first iteration of Seminaive. A major assumption is that G is connected. Otherwise, our analysis can be generalized to each connected component (subgraph) and the global time/cost is the sum of individual times/costs. We start with a minimally connected graph, which intuitively captures the best complexity case.

Lemma: Let G be a tree. Then $|E \bowtie E| = \Theta(n)$.

Proof: Since G is a tree $m = n - 1$. The main idea is that the best case is a balanced tree and the worst case is a list (chain). Without loss of generality assume G is a balanced binary tree and n is a power of 2. Then the number of paths of length 2 needs to exclude the leaves: $n/2$. Then the parents of the leaves are paths of length 1 and therefore should be excluded as well. Therefore, $|E \bowtie E| = n/2 - n/4 = \Theta(n)$. For the upper bound, assume G is a list, where each vertex (except the last one) has only one outcoming edge and no cycles. The number of paths of length 2 needs to exclude the last 2 vertices. Therefore, $|E \bowtie E| = n - 2 = \Theta(n)$.

Lemma: Let G be a complete graph without self-loops. Then $|E \bowtie E| = O(n^3)$

Proof: Intuitively, we need to count the number of paths of length 2. There are $n(n-1)$ pairs of vertices $(i, j), i, j \in 1 \ldots n$ Notice that since G is directed $(i, j) \neq (j, i)$. For each pair (i, j) there are $n - 2$ paths of length 2. Therefore, $|E \bowtie E| = n(n-1)(n-2) = O(n^3)$.

The previous result makes clear a complete graph is the worst case for $\bowtie$. Notice that if we make G cyclic by adding one edge so that $m = n$ $O()$ remains the same. Therefore, we propose the following result to characterize $O()$:

Proposition: Let H be a subgraph of G such that H is a complete subgraph (i.e. a clique). Let K be the number of vertices of H. By the previous lemma time is $O(K^2)$. Then the most important issue is K being independent from n, which leads to three cases: (1) If $K = \Theta(1)$ then H does not impact time. (2) if $K = \Theta(f(n))$ and $f(n) = o(n)$ then time is better than $\Theta(n^3)$. Prominent cases are $f(n) = \sqrt{n}$ or $f(n) = \log(n)$. (3) If $K = \Theta(n)$ then time is $\Theta(n^3)$. This result highlights cliques are the most important pattern impacting processing time.

Our ultimate goal is to understand $|R_k|$, where $R_k = E \bowtie E \bowtie \cdots \bowtie E$, multiplying E k times as $k \to n$. Notice that computing E^k is harder than G^+ because Seminaive can stop sooner to find G^+ (e.g. if G is complete). Our first time complexity result shows a list is the worst case for connected G. As explained above, the improved Seminaive algorithm computes the union of partial results at the end of the k iterations. Therefore, it would need $n - 1$ iterations. On the other hand, if we computed G^+ after every iteration we can stop at $k = n/2 = \Theta(n)$. In short, if G is a "chain" the number of iterations is $\Theta(n)$. Therefore, we conjecture that the second aspect impacting time of Seminaive is the diameter κ of G (i.e., it stops in time $O(\kappa)$). To conclude this section, we stress that the most important aspect is detecting if the graphs behind $E^2, E^3, \ldots, E^k$ become denser as the k iterations move forward. In fact, $|R_k|$ can grow exponentially as k grows. Therefore, if E^k is dense duplicate elimination is mandatory, which is an optimization studied in the next section.

To simplify analysis, we assume a worst case $|R_d| = O(m)$, which holds at low k values and it is a reasonable assumption on graphs with skewed vertex degree distribution (e.g. containing cliques). Then time complexity for the join operator can range from $O(m)$ to $O(m^2)$ per iteration. Since R_d is a temporary table we assume it is not indexed. On the other hand, since E is an input table and it is continuously used, we assume it is either sorted by the join column or indexed. During evaluation, R_d is sorted in some specific order depending on the join algorithm. At a high level, these are the most important join algorithms, from slowest to fastest: (1) nested loop join, whose worst time complexity is $O(m^2)$, but which can generally be reduced to $O(m \cdot \log(m))$ if R_d or E are sorted by the join column. (2) sort-merge join, whose worst case time complexity is $O(m \cdot \log(m))$, assuming either table R_d or E is sorted. (3) hash join, whose worst case time complexity can be $O(m^2)$ with skewed collisions, but which on average is $O(m)$ assuming selective keys and uniform key value distribution, which heavily depends on G structure and density. That is, it is not useful in dense graphs because many edges are hashed to the same bucket. (4) Finally, merge join is the most efficient algorithm, which basically skips the sorting phase and requires only scanning both tables. This is a remarkably fast algorithm, with time complexity $O(m)$, but which assumes both tables are sorted.

3.4 Parallel Processing

We assume there are P processors (nodes in a parallel cluster) in a distributed memory (shared-nothing) architecture, where processors communicate with each other via message passing. We consider both classes of graph algorithms introduced above. Recall $n = |V|$.

Parallel Matrix-Vector Multiplication

For matrix-vector multiplication the main issue is to partition E and S so that joins can be locally computed. There are two solutions: (1) Since $|S| = \Theta(1)$ or $|S| = \Theta(n)$ then S can be replicated across all nodes at low cost and in some cases it may be updated in RAM. (2) S can be partitioned by the vertex column to compute the join with E: $E.j$ and S is simply partitioned by $S.i$. We cannot guarantee an even distribution for E, but for S we actually can. Therefore, speedup will depend only on skewed degree vertices being in a subset of the nodes. Assuming $n \gg P$, having s high degree vertices and $s > P$ it is reasonable to assume such s vertices can be evenly distributed across the P nodes. For either solution, once the join result is ready the sum aggregation required by matrix-vector multiplication is computed locally with a local sort (perhaps with hashing in a best case), but without a parallel sort, resulting in optimal speedup going from $O(n/P \log(n/P))$ (sparse) to $O(n^2/P \log(n/P))$ (dense).

Parallel Matrix-Matrix Multiplication

For matrix-matrix multiplication the main issue is how to partition E to make multiplication faster, considering that the join operation accesses E in a different order from the previous iteration. If E is dense with $m = O(n^2)$ a partition by squared blocks is trivial if $n \gg P$, but large graphs are rarely dense. That is, E is generally sparse. Then for sparse graphs there are two major solutions: (1) partitioning edges by one vertex; (2) partitioning edges by their two vertices. Partitioning by vertex (i or j) will suffer when V degrees distribution is skewed: a few processors will contain most of the edges: parallel speedup will have a bottleneck. Hash-partitioning edges by both of their vertices can provide an even distribution, but in general the neighbors of a vertex will be assigned to a different processor, resulting in expensive data transfer during query processing. Therefore, this solution is detrimental to joins. The relative merits of each solution will depend on P, skewed degree distribution and network speed. On the other hand, the aggregation of the join result will require sorting in parallel by i, j, which can range from $O(n/P \log(n/P)))$ (sparse) to $O(n^3/P \log(n^3/P))$ (dense) assuming merge sort can evenly partition $E \bowtie E$ (a reasonable assumption for large n).

Notice the lower bounds match if $|S| = |E| = \Theta(n)$, but the upper bound for matrix-matrix multiplication has a much higher time complexity than matrix-vector multiplication when $|E| = \Theta(n^2)$.

4 Related Work

Research on analyzing graphs with relational queries has received little attention, compared to Big Data graph analytic systems (e.g. GraphX, Giraph, Neo4j). Important works include [9,11,12]. Reference [11] establishes a connection between graph analytics and SQL and justifies recursive queries, a classical problem, deserves to be revisited. Reference [9] shows a columnar DBMS is faster than graph analytic systems to compute some graph problems including reachability and PageRank. On the other hand, [12] compares different DBMS storage mechanisms to compute transitive closure and all-pairs shortest paths. This work shows columnar DBMSs have a performance advantage over rows and arrays. Neither work attempts to lay a theoretical foundation on the time and space complexity of relational queries to solve graph problems.

Being a classical topic in CS theory, research on recursive queries is extensive, especially with the Datalog language (which subsumes SQL) [1,3,8,13–16,18,20]. Comparatively, adapting such algorithms and optimizations to relational databases has received less attention [5,10,17]. Fundamental algorithms to evaluate recursive queries include Seminaïve [3] (the main algorithm used by us, linear number of iterations) and Logarithmic [17] (useful when most paths are long, logarithmic number of iterations). The Seminaïve algorithm solves the most general class of recursive queries, which eventually reach a fixpoint [2,3]. Both algorithms iterate joining the input table with itself until no rows are added to the global result table (i.e., the iteration reaches a fixpoint). There is an independent line of research on adapting graph algorithms to compute transitive closure problem in a database system [2,8]. The Direct algorithm [2] is an outstanding solution using in-place updates and bit operations in main memory instead of creating intermediate results on disk. Unfortunately, the Direct algorithm is incompatible with SQL query processing mechanisms and therefore has not made its way into relational DBMSs like Seminaïve.

5 Conclusions

We studied the evaluation of relational queries solving many fundamental graph problems iterating two forms of matrix multiplications: matrix-vector multiplication and matrix-matrix multiplication. Without loss of generality we focused on directed graphs. We studied two major aspects: (1) time and space complexity, considering basic relational operators (select, project, join) and (2) parallel processing, analyzing speedup and issues with unbalanced data partitioning. Parallel processing requires two major steps at each iteration: a parallel join and a parallel group-by agregation. Our results highlight a relational join is the most demanding operator, followed by group-by aggregations, expressed as an extended projection operator. Time complexity per iteration is best when the graph is sparse and its shape resembles a balanced tree and it is worst when the input graph is very dense (i.e. $m = O(n^2)$) or when intermediate tables gradually approximate complete subgraphs (i.e. subgraphs embedded in G become denser

after each iteration). Parallel speedup is heavily impacted by a few vertices having a skewed degree distribution.

There are many open research issues, bridging theory and systems research. It is necessary to study graphs that have embedded cliques (complete subgraphs) in more depth. We need to study cardinality estimation of intermediate query results considering the graph is a union of subgraphs of diverse structure and density. Assuming the input graph is sparse but its transitive closure at some recursion depth is dense we need to characterize more precisely when a hash join or a sort-merge join is preferable. Finally, we aim to study harder graph problems requiring non-linear recursion (i.e. clique detection), which can help us identify SQL extensions to make it as powerful as Datalog.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases : The Logical Level, Facsimile edn. Pearson Education POD, London (1994)
2. Agrawal, R., Dar, S., Jagadish, H.V.: Direct and transitive closure algorithms: design and performance evaluation. ACM TODS **15**(3), 427–458 (1990)
3. Bancilhon, F., Ramakrishnan, R.: An amateur's introduction to recursive query processing strategies. In: Proceedings of ACM SIGMOD Conference, pp. 16–52 (1986)
4. Cabrera, W., Ordonez, C.: Scalable parallel graph algorithms with matrix-vector multiplication evaluated with queries. Distrib. Parallel Databases **35**(3–4), 335–362 (2017)
5. Dar, S., Agrawal, R.: Extending SQL with generalized transitive closure. IEEE Trans. Knowl. Eng. **5**(5), 799–812 (1993)
6. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book, 1st edn. Prentice Hall, Upper Saddle River (2001)
7. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning, 1st edn. Springer, New York (2001). https://doi.org/10.1007/978-0-387-21606-5
8. Ioannidis, Y.E., Ramakrishnan, R., Winger, L.: Transitive closure algorithms based on graph traversal. ACM TODS **18**(3), 512–576 (1993)
9. Jindal, A., Rawlani, P., Wu, E., Madden, S., Deshpande, A., Stonebraker, M.: Vertexica: your relational friend for graph analytics! Proc. VLDB Endow. **7**(13), 1669–1672 (2014)
10. Mumick, I.S., Finkelstein, S.J., Pirahesh, H., Ramakrishnan, R.: Magic conditions. ACM TODS **21**(1), 107–155 (1996)
11. Ordonez, C.: Optimization of linear recursive queries in SQL. IEEE Trans. Knowl. Data Eng. (TKDE) **22**(2), 264–277 (2010)
12. Ordonez, C., Cabrera, W., Gurram, A.: Comparing columnar, row and array DBMSs to process recursive queries on graphs. Inf. Syst. **63**, 66–79 (2017)
13. Ramakrishnan, R., Srivastava, D., Sudarshan, S., Seshadri, P.: Implementation of the CORAL deductive database system. In: Proceedings of ACM SIGMOD, pp. 167–176 (1993)
14. Sippu, S., Soininen, E.S.: An analysis of magic sets and related optimization strategies for logic queries. J. ACM **43**(6), 1046–1088 (1996)
15. Sakr, S., Elnikety, S., He, Y.: Hybrid query execution engine for large attributed graphs. Inf. Syst. **41**, 45–73 (2014)

16. Ullman, J.D.: Implementation of logical query languages for databases. ACM Trans. Database Syst. **10**(3), 289–321 (1985)
17. Valduriez, P., Boral, H.: Evaluation of recursive queries using join indices. In: Expert Database Systems, pp. 271–293 (1986)
18. Vardi, M.Y.: Decidability and undecidability results for boundedness of linear recursive queries. In: ACM PODS Conference, pp. 341–351 (1988)
19. Yakovets, N., Godfrey, P., Gryz, J.: Evaluation of SPARQL property paths via recursive SQL. In: Proceedings of AMW (2013)
20. Youn, C., Kim, H., Henschen, L.J., Han, J.: Classification and compilation of linear recursive queries in deductive databases. IEEE TKDE **4**(1), 52–67 (1992)

Using Functional Dependencies in Conversion of Relational Databases to Graph Databases

Youmna A. Megid(✉), Neamat El-Tazi, and Aly Fahmy

Faculty of Computers and Information, Cairo University, Giza, Egypt
YoumnaAbdElmegid@gmail.com,
{n.eltazi,a.fahmy}@fci-cu.edu.eg

Abstract. Graph database management systems are widely used in scenarios where the data are intensively connected. Handling such connected data in a relational database is not an efficient task. Converting relational databases to graph ones is one of the solutions that can empower users with handling such data using the graph model features. In this paper, we propose a new algorithm to ease such conversion and overcome the limitations of the existing algorithms. The state of the art algorithms cannot handle multiple relationships types such as unary relationships and associative entities with non-foreign key attributes. Our proposed algorithm, FD2G, leverages the existence of functional dependencies information inside the input relational database to automatically perform the conversion to property graph databases. In addition, we updated the state of the art algorithm, named R2G, to handle its limitations and be able to fairly compare both algorithms performance. We evaluated FD2G against the updated R2G algorithm where it efficiently and effectively outperformed the existing one.

Keywords: Graph databases · Relational models · Functional dependencies

1 Introduction

Social media applications nowadays result in an exponential growth of interconnected data. Research studies [1, 2] have been conducted to convert relational databases to graph databases to benefit from the efficiency of query processing when the data of such applications is extensively connected. R2G algorithm [1] was proposed to convert relational to graph databases, migrate the data and evaluate the relational queries over the graph queries. Applying R2G algorithm to the same database provided in the authors' paper [1] resulted in a different output graph database.

In this paper, we modified R2G algorithm to reach the same output graph in paper [1] to fairly compare our new proposed algorithm to the modified R2G algorithm.

Consequently, we propose a new algorithm for the conversion that takes into consideration all the uncovered cases of R2G. Our proposed algorithm leverages the information in the functional dependencies to generate a converted graph database that has better performance in answering queries against the state of the art algorithms.

The rest of this paper is organized as follows: Sect. 2 presents the limitation in R2G algorithm. Section 3 shows our updates in R2G algorithm. Our proposed algorithm is

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 350–357, 2018.
https://doi.org/10.1007/978-3-319-98812-2_31

presented in Sect. 4. Section 5 evaluates the performance of the proposed algorithm against the updated R2G algorithm. Finally, Sect. 6 concludes the paper.

2 Related Work

Existing researchers [1–3] studied the conversion of relational databases to graph databases. One of the limitations of R2G algorithm [1, 2] is that unary relationships are not covered. Using the schema graph, full schema paths are generated starting from each source node until any sink node as denoted in R2G. Adding a unary relationship to the schema graph will lead to an infinite number of schema paths since a cycle will be created in the schema graph.

In addition, if the relational database has redundant attributes in one of the relations, when the data is converted to graph database, data will be lost during the conversion process. R2G creates a node type for each entity and inserts all related data inside it. For instance, consider the database in Fig. 4, with orders and order items details, the order will be added in one node and order items details will be replacing each other's values resulting in only one order item detail inside the node losing the data of the other order items.

3 Updated R2G

After applying R2G algorithms as is to the input database mentioned in the R2G paper [1], the result will be the output graph database in Fig. 1a which is not the same as the one presented in the authors paper (here in Fig. 1b). Figure 1b is the correct one according to the R2G authors, but the authors did not take into consideration the unary relationships and we argue that it is not possible to generate such correct output without handing the missing cases.

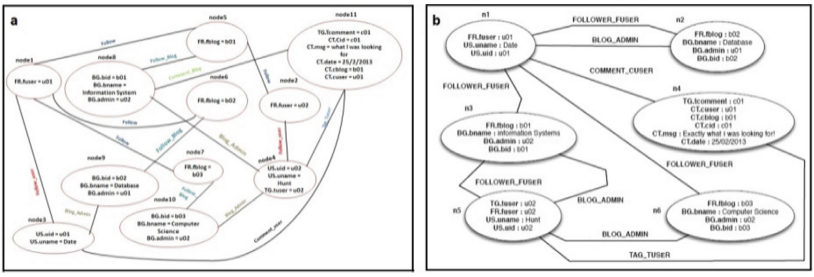


Fig. 1. a - Output graph after applying R2G as is. b - Output graph in R2G presented in [1]

By comparing the two graphs a and b in Fig. 1, multiple differences are detected. Number of nodes in Fig. 1a is larger than number of nodes in Fig. 1b. The reason for that is properties are distributed over multiple nodes in Fig. 1a. However, in Fig. 1b all

properties are consolidated in one node. For instance, properties in node n1 of Fig. 1b is distributed on nodes node1 and node3 in Fig. 1a.

Moreover, there exist differences in the number and types of edges between nodes. For instance, Fig. 1a has 15 edges with 7 different types, while Fig. 1b has only 10 edges with 5 different types.

In addition, there is a conflict between case 4 and case 5 discussed in [1]. If the current attribute is n2n and not visited before, both case 4 and case 5 can be applied but each case will generate different number of nodes in the output graph.

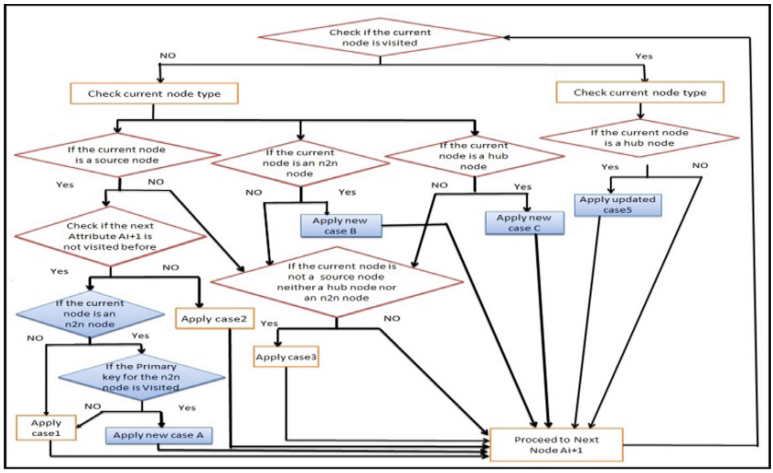


Fig. 2. Conditions for applying updated R2G cases

Figure 2 presents the conditions of applying the new cases that were introduced to reach to the output in Fig. 1b. The rest of the section illustrates the updates in details.

3.1 New: Case A

As presented in Fig. 2, this case applied if the current attribute A_i and the A_{i+1} are not visited. Current attribute should be n2n node and the primary key of A_i is visited before. All nodes containing the primary key of the current attribute A_i should be retrieved. These nodes will be updated by inserting the A_i data values as properties to this node and A_i should be added to the visited list.

3.2 New: Case B

The conditions for Case 4 in R2G were divided into two conditions with two resulting new cases (case B and case C).

Retrieve the primary key attribute for the current n2n attribute ($A_{i\text{-}pk}$). If $A_{i\text{-}pk}$ was not visited before, for each attribute value of the current attribute A_i, a new node should be created and the attribute value will be added as a property to the created node.

On the other hand, if $A_{i\text{-}pk}$ was already visited, all nodes created or updated by $A_{i\text{-}pk}$ should be retrieved with their values ($A_{i\text{-}pk}$, V_2). Since we are following foreign key relations between A_i and $A_{i\text{-}pk}$, the nodes retrieved will be updated by adding the attribute value of A_i (V_1) as a new property (such that $V_1 = V_2$).

All nodes created or updated by the previous attribute $A_{i\text{-}1}$ should be retrieved afterwards to check their data since A_i and $A_{i\text{-}1}$ are attributes of the same relation; consequently, data of the same tuple are followed. A new edge should be created between nodes containing A_i and nodes containing $A_{i\text{-}1}$ for each tuple. Finally, A_i will be added to the visited list.

3.3 New: Case C

Retrieve all nodes created or updated by the previous attribute $A_{i\text{-}1}$. In case $A_{i\text{-}1}$ is an n2n node, meaning that there is a foreign key relation between the current attribute A_i and $A_{i\text{-}1}$, the nodes retrieved must be updated by inserting the current attribute values as properties to these nodes. However, in case that there is no node containing the attribute value for $A_{i\text{-}1}$, a new node will be created for each attribute values for A_i, and the attribute value will be added as a property for the created nodes.

If $A_{i\text{-}1}$ is not an n2n node, and a foreign key relationship exists between a visited n2n node A_X (assume A_X attribute value is V_2) and A_i (assume A_i attribute value is V_1), The new property (A_i, V_1) has to be added to the node that contains (A_X, V_2) such that $V_1 = V_2$. If there is no satisfying now with that condition ($V_1 = V_2$), a new node has to be created and the property (A_i, V_1) should be inserted inside it.

Finally, a new edge between the nodes updated by the current attribute A_i and the nodes created by the previous attribute $A_{i\text{-}1}$ should be created and A_i should be added to the visited list.

3.4 Updated: Case 5

To apply this case, iterate on all nodes updated by the previous attribute $A_{i\text{-}1}$ and nodes updated by the current attribute A_i. In case that the two attributes exist in two different nodes, a new edge should be created between these two nodes. according to the foreign relation that exists between A_i and $A_{i\text{-}1}$.

4 Fd2G Conversion Algorithm

The inputs of the FD2G algorithm are a relational database and a list of functional dependencies (if exists) or generated automatically by the algorithm (if does not exist). The output is property graph database model and migrated data.

FD2G main idea is to convert each functional dependency to a new node type and all attributes for the functional dependency will be added as properties for this new node type. Except when the functional dependency represents many-to-many relationship, it will be divided into two cases.

Case 1: is applied when all attributes in the functional dependency are foreign keys. Edges will be created between all the left-hand side attributes of the functional dependency

Case 2: is applied when the functional dependency contains non-foreign keys attributes. Edges will be created between all the left-hand side attributes of the functional dependency and all right hand-side attributes will be added as properties to these edges.

All other relationships in the input relational database will be simply converted to edges between nodes.

```
Algorithm 1: Convert Relational Database RDB to Graph Database GDB
Input: Relational Database RDB and Functional dependencies FDList
Output: Graph database GDB
Begin Algorithm
   NormalizeRelationalDB(RDB);
   RelationsList //Load all FK Relations from RDB;
   For each FD in FDList //Set of functional dependencies in RDB
      If (FD not represents many-To-many relationship) then
         NodeType = RelTableName;
         RDBTable = GetRelationalTable(RelTableName, RDB);
         For each Row in RDBTable
              NewNode = GDB.createNode(NodeType);
              For each ColumnName, AttributeVal in Row
                   NewNode.setProperty (ColumnName, AttributeVal);
              End For
         End For
      End If
   End for
   For each FD2 in FDList
         If (FD2 represents many-to-many relationship) then
            If (FD2 contains non-foreign keys attributes) then
               CreateEdgeWithProperties(FD, GDB, RDB);
            Else
               CreateEdgeWithoutProperties(FD, GDB, RDB);
         End If
   End For
   For each Rel in RelationsList
         If (Rel not exist in GDB) then
              CreateNewEdge(Rel, GDB, RDB);
   End for
End of Algorithm
```

Fig. 3. Algorithm 1 pseudocode

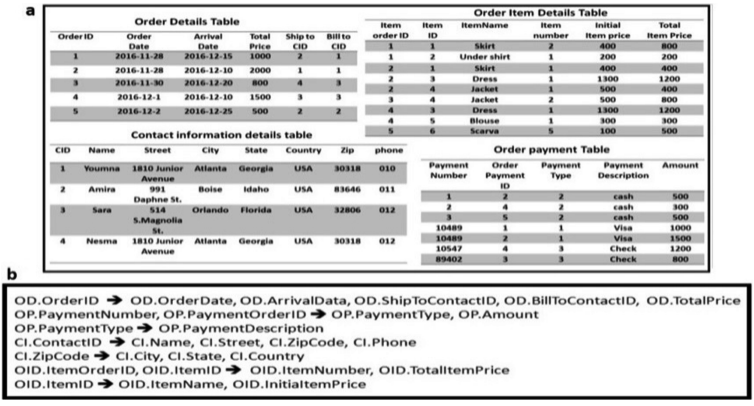


Fig. 4. a - Order store relational DB. b - Functional dependency for order store DB

Consider the database in Fig. 4a and the functional dependencies in Fig. 4b, by applying FD2G algorithm represented in Fig. 3, the output graph database contains six different node types as presented in Fig. 5 with each node type in different color.

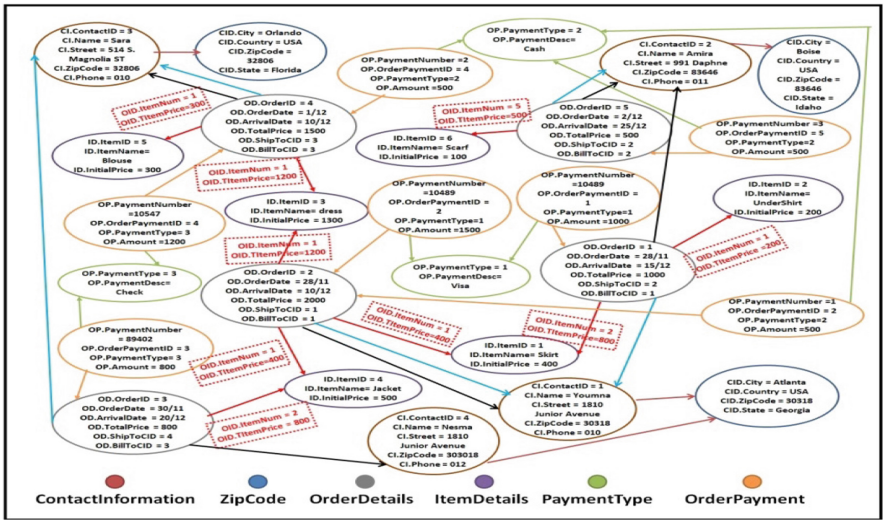


Fig. 5. Output graph database for order store database (Color figure online)

5 Experimental Results

The proposed algorithm FD2G was evaluated against the updated R2G algorithm. Both algorithms were implemented using Java, SQL server and Neo4j database management systems. Experiments were conducted on core i7-6700HQ, 2.60 GHZ running Windows 7 with 8 GB of memory.

5.1 Datasets

Datasets in [1, 4, 5] were used in the evaluation. All the datasets are presented in in at Fig. 6a with their characteristics as number of tables, tuples and relationships.

a

Dataset	Tables	Tuples	Relationships
Order store database [Fig. 4]	4	25	4
Relational database R2G [1]	5	11	7
North-wind database [4]	13	3,308	13
Wikipedia-2008 subset database [5]	6	200,000	6
IMDB subset Database [5]	6	1,673,074	5

b

Databases	FD2G	Updated R2G
Order store database	18.005 sec	39.587 sec
Relational database in R2G	16.698 sec	19.269 sec
North-wind database	1 min 18.027 sec	Not supported resulting in a cycle in the schema graph with no output.
Wikipedia 2008 subset database	27 min and 47.285 sec	2 hrs 57 min 47.684 sec
IMDB database	95 hrs, 15 min & 07.9 sec	>300 hrs

c

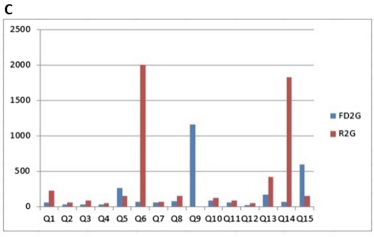


Fig. 6. Experimental results for comparing updated R2G with FD2G

We excluded only Mondial dataset from datasets in [5], as there were no defined foreign keys inside the database structure, which will make it hard to deduce the relationships that FD2G needs in conversion.

5.2 Conversion Time

Conversion time of each dataset using the two algorithms is represented in Fig. 6b. FD2G outperforms the updated R2G, as R2G algorithm depends on the full schema paths generated from the relational database and hence the conversion process is applied for each attribute separately. On the other hand, the FD2G algorithm conversion process is applied for each functional dependency with all its attributes.

5.3 Query Processing Time

Query processing time was evaluated by applying the same set of queries presented in Fig. 7 on the Wikipedia relational database. The results of processing the queries are presented on the same figure. From the results, the transformed data using FD2G algorithm is complete when mapping it with the relational results.

As presented in Fig. 6c most queries result in a better evaluation time using FD2G. On the other hand, R2G graph database showed a better performance in two queries, Q5 and Q15. From Fig. 7, both queries results in only one node in R2G while it results in different number of rows and nodes in the relational database and FD2G. For instance, Q5 results in 1800 nodes using FD2G and 1800 rows using the relational database while only one node with updated R2G. This is because in the updated R2G graph database, the page-link data is stored in the node that contains the page data. Each new page-link data for the same page will replace the old page-link data resulting in missing data and a wrong query output.

Q	Query Statement	Relational	FD2G	Updated R2G
Q1	Retrieve the user name that review page "Qing_Dynasty"	1 row	1 node	1 node
Q2	Retrieve all pages reviewed by user "Andrew Nixon"	1 row	1 node	1 node
Q3	Retrieve all pages reviewed by user "Addshore"	8 rows	8 nodes	8 nodes
Q4	Retrieve review comments on page "War_in_Somalia_(2006–present)"	1 row	1 node	1 node
Q5	Retrieve all pagelinks where pl_to = 2385	1800 rows	1800 nodes	1 node
Q6	Retrieve all pages reviewed where review length > 100000	136 rows	136 nodes	136 nodes
Q7	Retrieve list of all users reviewed pages such that page length >10000	55 rows	55 nodes	55 nodes
Q8	Retrieve users that review more than 9 page	43 rows	43 nodes	43 nodes
Q9	Retrieve all pages reviewed by users	5540 rows	5540 nodes	Crashed with large num. of returned nodes
Q10	Retrieve users that review more than 50 page	5 rows	5 nodes	5 nodes
Q11	Retrieve page contents for all reviewed pages where page_len > 200000	1 row	1 node	1 node
Q12	Retrieve all pages where page length > 100000 page	136 rows	136 nodes	136 nodes
Q13	Retrieve all page contents reviewed by user id (1309)	19 rows	19 nodes	19 nodes
Q14	Retrieve all users review pages where page_len > 100000	136 rows	136 nodes	136 nodes
Q15	Retrieve all pagelinks where page Title contain Silver	145 rows	145 nodes	1 node

Fig. 7. Wikipedia queries used in testing phase

When executing Q9 on the updated R2G graph database, the results could not be retrieved due to the inability of R2G algorithm to handle large number of nodes, while in FD2G database, the query was executed, and its result shows the whole graph that represents the query result. Figure 6c shows the difference between the query processing times of the 15 queries of Fig. 7.

6 Conclusion

In this paper, we proposed a new algorithm, FD2G, for converting and migrating a relational database to a property graph database. Updates on the state of the art algorithm, R2G, were also proposed to overcome its limitations. An evaluation of the conversion on both algorithms showed that the new proposed algorithm FD2G outperformed R2G in handling multiple relationships types such as unary relationships and associative entities with non-foreign key attributes. Unlike R2G algorithm which loses data during the conversion of an un-normalized relational database, FD2G normalizes the relational database to third normal form first before the conversion process resulting in a complete graph database without losing data. Moreover, processing query times were evaluated showing that FD2G output graph model handles queries more efficiently than R2G algorithm.

References

1. De Virgilio, R., Maccioni, A., Torlone, R.: Converting relational to graph databases. In: Proceedings of the First International Workshop on Graph Data Management Experience and Systems, New York, NY, USA, pp. 1–6, June 2013. https://doi.org/10.1145/2484425.2484426
2. De Virgilio, R., Maccioni, A., Torlone, R.: R2G: a tool for migrating relations to graphs. In: Proceeding of the 17th International Conference on Extending Database Technology, EDBT/ICDT 2014 Joint Conference, Athens, Greece, pp. 640–643, March 2014. https://doi.org/10.5441/002/edbt.2014.63
3. Gupta, M., Rani Aggarwal, R.: Transforming relational database to graph database using Neo4j. In: Proceedings of the Second International Conference on Emerging Research in Computing, Information, Communication and Applications, Bangalore, India, pp. 322–331. ELSEVIER Publications (2014)
4. Downloading sample databases – North Wind database. https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/linq/downloading-sample-databases. Accessed 27 Mar 2018
5. Coffman, J., Weaver, A.C.: A framework for evaluating database keyword search strategies. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, pp. 729–738, October 2010. https://doi.org/10.1145/1871437.1871531

Learning

A Two-Level Attentive Pooling Based Hybrid Network for Question Answer Matching Task

Zhenhua Huang[1,2], Guangxu Shan[2], Jiujun Cheng[2], and Juan Ni[1(✉)]

[1] South China Normal University, Guangzhou 510631, China
jukiehuang@163.com, nijuanshkj@126.com
[2] Tongji University, Shanghai 201804, China

Abstract. Attention-based deep learning network models have shown obvious advantages on the sentence representation in many NLP tasks. While in the answer selection domain, applying attention-based deep learning model to capture complex semantic relations between question and answer is an extremely challenging task. In this paper, instead of simply using max-pooling in the pooling layer, we propose the two-level attentive pooling model which can efficiently select several key and high semantic-related matching words in the question-answer pair to improve the accuracy of answer selection. Specially, our model is built on top of the hybrid network which includes GRU and CNN to encode the complex sentence representation. The experimental evaluation on two popular datasets shows that our model has the good effectiveness and achieves the state-of-art performance in the answer selection task.

Keywords: Two-level attention pooling · Hybrid neural network
Answer selection · Deep learning

1 Introduction

Recently, deep neural networks have achieved great results in many NLP tasks due to its ability to learn more abstract and advanced features [1]. Applying deep neural networks to question-answer matching task has made great progress in recent years [2]. A robust semantic matching model of question-answer pair considers not only the internal syntactic structure of a sentence but also some key words that have important semantic features. A usual practice is to use the attentive model with considering the interdependence between question-answer pair to achieve this target.

Inspired by the two-way attention mechanism [3], there are several attention models [4, 5] aiming to learn a two-way soft alignment matrix between question-answer pair. This matrix represents the similarity score between question-answer pair word-by-word and then implements the max pooling (column & row) operation. However, this simple pooling mechanism will lose some other important feature information by only selecting the max value. Moreover, the max-pooling mechanism will lose the position information of word order.

In order to obtain key segments to learn more composite features which can be efficiently used to select the ground-truth answer and reduce the influence of noise words, in this paper, we propose a two-level attentive pooling-way network model for

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 361–368, 2018.
https://doi.org/10.1007/978-3-319-98812-2_32

the question-answer matching task. Instead of only utilizing the max-pooling, we use the max-pooling at the first step and then use the max-K pooling at the second step to select key matching words on the intermediate matrix. Furthermore, we also use the pre-trained word vector to improve the final result. We mainly report experimental results on the two different datasets: NLPCC-ICCPOL 2016 Shared Task on DBQA [6] and Insurance QA [7]. The experimental results show that our model is better than the strong baselines and achieves the state-of-art performance.

2 Two-Level Attentive Pooling Network Model

2.1 Word Embedding Construction

Given $Q = \{q_1, q_2, \ldots, q_{L1}\}$ and $A = \{a_1, a_2, \ldots, a_{L2}\}$ represent the words in the question and answer, respectively. Then based on *word2vec* [8], the outputs are two distributed matrices: $Q \in R^{d \times L_1}$ for the question and $A \in R^{d \times L_2}$ for the answer.

2.2 Bi-GRU Encoding and Convolution Encoding

In the first layer of our model, we adopt Bi-GRU [9] which utilizes both the previous and future contexts by processing the sequence on two directions.

Given an input sequence $X = \{X_1, \ldots, X_L\}$, $X_t \in R^d$ at the position of t. Then the output of this layer is the concertation of both directions of hidden state as follows:

$$E(X) = Q(X)|A(X) \tag{1}$$

$$\overrightarrow{H_t} = \overrightarrow{GRU}(E(X)) \tag{2}$$

$$\overleftarrow{H_t} = \overleftarrow{GRU}(E(X)) \tag{3}$$

$$H_t = \overrightarrow{H_t} || \overleftarrow{H_t} \tag{4}$$

where "||" is the concatenation operation. We define $\mathbb{N} = 2 \times h$. Hence after computing the hidden state $H_t \in R^h$ for each time step t, we generate the matrices $Q \in R^{\mathbb{N} \times L_1}$ for the questions and $A \in R^{\mathbb{N} \times L_2}$ for the answers respectively. And the j-th column in Q and A represents the corresponding j-th hidden state H_j computed by the Bi-GRU.

In the second layer of our model, we use CNNs [10] to obtain the local n-gram coherence interacted with each other in the sentence. The convolutions structure adopted in our model can be described as follows: (1) Firstly, given the words sequence $q^{gru} = \{q_1, q_2, \ldots q_L\}$, $q_t \in R^{\mathbb{N}}$, we define a matrix $S \in R^{k\mathbb{N} \times L}$, where L is the sequence-length. Then the m-th column of S, i.e. $S_m \in R^{k\mathbb{N}}$, is the concatenation of a sequence of k word-embedding centralized in the m-th word of q^{gru}. In our model, the output of convolution over the words sequence can be calculated as follows:

$$G = relu(W_{gc}S + b) \tag{5}$$

where $relu(\cdot)$ is the non-linearity activation function instead of $tanh(\cdot)$, b is the bias vector and W_{gc} represents the weight matrix. (2) Secondly, we further get the feature matrix $G \in R^{C \times L}$ in which each column contains the features extracted from k window-size context of the words sequence, where C is the filter numbers in convolution and k is hyper-parameter can be chosen on the validation set.

After the convolution, we can obtain the matrix $Q \in R^{C \times L_1}$ for questions and $A \in R^{C \times L_2}$ for candidate answers respectively.

2.3 Two-Level Attentive Pooling Architecture

Inspired by the literature [3], we compute a soft alignment matrix $D \in R^{L_1 \times L_2}$ between the question-answer pair by comparing each word in the sentence of question and answer. In our model, we use a cosine distance function to compute D. Then we use the first-level max-pooling [11] which only chooses the max matching degree of a row or column in D between the question-answer pair. The max matching degree represents the importance score for a k-size window context centralized the j-th word in one sentence with regard to another entire sentence.

In the first step we adopt max-pooling on each row and column of D. And we can obtain the pooled vectors for the question $q \in R^{L1}$ and the answer $a \in R^{L2}$. In order to keep some main higher alignment words in a sentence which are dominantly contribute to the final result and can reduce the influence of noisy words, we implement max-K pooling over the first pooled vector q and a. We only select the key K words. We call this pooling strategy as *two-level attentive pooling network*.

Then, we employ these K key word-segments as the input of next RNN layer to encode the sequential words. In this layer, we use the GRU model. Differently from the first Bi-GRU layer, we only select the last hidden state which contains all the previous information as the representation of K *key-matching* words in the sentence. We denote the output of feature vector from this RNN encoding layer of question and answer as O_q and O_a, respectively. Before calculating the matching similarity, we execute the dropout operation on the question and answer vector representation.

2.4 Final Feature Vector

After obtaining the last output from GRU model, we can use it to the main feature words and compute the question-answer matching score. we follow the approach [12] that define the similarity between the vectors O_q and O_a as follows:

$$simi(O_q, O_a) = O_q^T M O_a \tag{6}$$

The matrix M transforms the candidate answer O_a to the closest input question O_q. And then, we concatenate all the intermediate vectors, the question feature vector O_q, the answer feature vector O_a and their matching score $simi(O_q, O_a)$ into a single vector X, and feed it to a fully connected hidden layer which allows for modeling interactions

among the components of joint vector. Finally, we put the output of hidden layer into the *softmax* layer as follows:

$$p = \text{softmax}(W_h x + b) \tag{7}$$

Specially, the model is trained to minimize the cross-entropy cost function:

$$Lost = -\sum_{i=1}^{N} \{y_i \log p_i + (1 - y_i) \log(1 - p_i)\} \tag{8}$$

where p_i is the result of the logistic layer. We use the L_2 norm regulation to avoid overfitting in training.

Figure 1 illustrates the framework of our model.

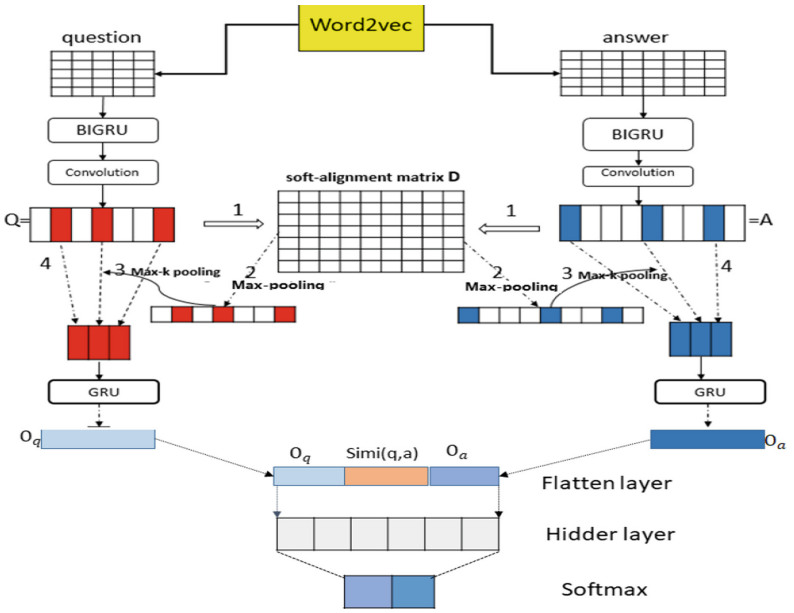


Fig. 1. The two-level attentive pooling network model for answer selection task

3 Experimental Setup

3.1 Datasets

We conduct the experimental evaluation on two prevalent datasets: NLPCC-ICCPOL 2016 Shared Task on DBQA [6] and Insurance QA [7]. The good experimental results on these two datasets demonstrate that the robustness and stability of our proposed model.

(1) NLPCC-ICCPOL 2016 Shared Task on DBQA: this dataset is divided into the train set and the test set. The train set contains 181882 question-answer pairs and 8772 questions, and the test set contains 122531 question-answer pairs and 5997 questions. We first use an open-source participle tool *pynlpir* [13] segment each whole sentence. After segmentation, we can find that the average length of questions is less than 8, while the average length of answers is more than 90. In the training, we split 0.1% of the train set as the validation set (dev) to tune hyper-parameters.
(2) Insurance QA: it is a domain-specific answer selection dataset, hence the sentence contains some domain-key words which may largely determine the right answer. The vocabulary size of the Insurance QA is 22353. And the average length of questions in this dataset is less than 20, while the average length of answers in this dataset is more than 39.

The statistic of questions and answers on these two datasets is shown in Table 1.

Table 1. The statistic of questions and answers in two datasets, L_1 is the average length of questions, L_2 is the average length of answers

Datasets	Train	Dev	Test	L_1	L_2
Chinese DBQA	12887	1000	2 * 1800	7.16	94.6
Insurance QA	8772	1000	2779	19.65	39.1

3.2 Hybrid Network Setup

For the Chinese Open-Domain Question-Answering task dataset, we first use pynlpir to segment the question and answer sentences respectively due to lacking of obvious words boundaries of Chinese sentence, and then use the pre-trained *word2vec* to obtain word embedding on *Baidu Encyclopedia corpus*. The pre-trained word embedding used as the original input of neural network is very import to results.

In the training set, we remove those questions without any labeled correct answer and then add wrong-answers which are randomly selected from answers-pool for questions that only have one right answer. We also add the word overlap and tagging feature into the dimension of our pre-train word vector. Moreover, we remove some meaningless stop words. The word-vector size is set to 156. And the *Adam* algorithm is used as the optimizer with the first momentum coefficient of 0.9 and the second momentum coefficient of 0.99. The mini-batch size is set to 64 and we set the context window-size k = 3, 4, 5 in the convolution layer with the filter number of 64. The pooling K hyper parameter for this dataset is set to 6, which can achieve the best results in our model.

For the Insurance QA dataset, the training data for the word embedding is a *Wikipedia corpus* which contains 164 million question-answer pairs. In order to compare with other existing models on this dataset, we set the vector-size to 100 and set the mini-batch size to 20. Similar with [10], we set the size of candidate answers pool to 100. The candidate answers pool includes ground-truth answers and randomly selected negative answers for each question on the validation set and two test sets.

We also set the window-size k = 3, 4, 5 to extract different feature informations in the convolution layer. The best hyper parameter of K is set to 5 for the attentive K-max pooling, the hidden size of first Bi-GRU layer is set to 50, and the hidden size of second GRU layer is also set to 50. All experiments are processed with the GPU on the server with *Cuda 7.5* driver supported by NVIDIA.

4 Experiments Results

4.1 The Chinese Open-Domain Question-Answering Task Dataset

The Chinese Open-Domain Question-Answering task can be regarded as a ranking problem, and therefore we can utilize mean reciprocal rank (MRR) and mean average precision (MAP) as the evaluation metrics. The experiment results are reported in Table 2.

Table 2. Evaluation of different methods on the Chinese Open-Domain Question-Answering task dataset

	Method	MRR	MAP
Baseline	Bag-of-words	30.22	30.56
Literature [14]	CNN based	36.42	36.41
	CNN + interact	59.21	59.14
	CNN + interact + overlap	85.92	85.86
	AP based-CNN	84.96	84.90
Our model	Two attentive pooling + (K = 5)	85.34	**85.94**
	Two attentive pooling + (K = 6)	**86.01**	85.80

For comparisons, we report the performance on the baseline *bag-of-words* (BOW) method which uses the *idf-weighted* sum of word vectors as the final feature vectors to match the question and each of its candidate answers. To the best of knowledge, the literature [14] is the best one among existing methods on the Chinese Open-Domain Question-Answering task dataset used in the NLPCC-ICCPOL 2016 shared task.

In experiments, we can easily find that the BOW model cannot learn the deeper features and has the worst experimental performance, and our model are better than the CNN-based models and its variations in the literature [14]. Specially, from Table 2, we can observe that attention-based deep learning models can obviously improve the experimental results compared with separate deep neural networks. Moreover, different values of the hyper-parameter K influence the experimental results on the test set. We attempt to test different values of K and find that when K = 5 or 6 it has the best performance and starts to drop when $K > 6$.

In addition, we can further find that our model which use two-level pooling (i.e., one is the max-pooling, another one is the max-K pooling) can efficiently select best key words of semantic-similarity on question-answer pair as the final feature vectors when there exists many unrelated words in Chinese sentences.

4.2 The Insurance QA Dataset

In Table 3, we present the experimental results for different models on the Insurance QA dataset, and the results are evaluated by accuracy. The CNN-based model proposed in [10] has no attention mechanism and its accuracy is inferior to the LSTM-based model with attention mechanism [15] from question to answer. However, all these two models perform much better than the traditional BOW model. The AP-CNN and AP-BiLSTM models proposed in [11] use two-way attention to learn semantic similarity between words for each question-answer pair, and hence perform better than both of the CNN-based model and the LSTM-based model.

On the other hand, we can observe that our model are better than the AP-CNN and AP-BiLSTM models. The main reason is that differenced with these two models, our model has two-level pooling steps: (1) the first max-pooling step is to calculate the semantics matching degree for a word in question with regard to the other word in answer side by generating the alignment matrix D, and (2) the second max-K pooling step mainly remove some noisy words and only keep the features of key words which can determine the final result. Meanwhile, Table 3 indicates that adding the second max-K attention pooling step can perform better than only one attention-pooling step which is used in existing models.

Table 3. Evaluation of accuracy in the insurance QA dataset for different models

	Method	Dev	Test1	Test2
Baseline	Bag-of-words	31.9	32.1	32.2
Literature [14]	CNN [10]	65.4	65.3	61.0
	Attention-LSTM [3]	66.5	63.7	60.3
	AP-CNN [11]	68.8	69.8	66.3
	AP-BiLSTM [11]	68.4	71.7	66.4
Our model	Two attentive pooling + (K = 5)	**68.9**	69.9	66.3
	Two attentive pooling + (K = 6)	68.5	**72.4**	**67.1**

5 Conclusions

In this paper, we propose the two-level attentive pooling network model on the question-answer matching task. The model is built on top of the Bi-GRU and CNN architectures. In the first step, we utilize the max-pooling to get the scoring vector. Then in the second step, we use the max-K pooling for the pooled vector which is obtained in the first step to remove noisy segments in a sentence. We conduct experiments on the two different language datasets (i.e., Chinese and English). The experimental results show that our model outperforms the strong baselines and achieves the state of art performance.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (61772366), the Natural Science Foundation of Shanghai (17ZR1445900) and the Fundamental Research Funds for the Central Universities.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXivpreprint arXiv:1409.0473, pp. 1–15 (2014)
2. Chung, J., Gulcehre, C., Cho, K.H., et al.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, pp. 1–9 (2014)
3. Cui, Y., Chen, Z., Wei, S., et al.: Attention-over-attention neural networks for reading comprehension. arXiv preprint arXiv:1607.04423, pp. 1–10 (2016)
4. Zhang, J., Zhu, X., Chen, Q., et al.: Exploring question understanding and adaptation in neural-network-based question answering. arXiv preprint arXiv:1703.04617, pp. 1–11 (2017)
5. Yang, L., Ai, Q., Guo, J., et al.: ANMM: ranking short answer texts with attention-based neural matching model. In: 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, pp. 927–935. ACM (2016)
6. NLPCC2016-DBQA-DATA. http://pan.baidu.com/s/1c138KZQ. Accessed 21 May 2018
7. Hermann, K.M., Kočiský, T., Grefenstette, E., et al.: Teaching machines to read and comprehend. In: International Conference on Advances in Neural Information Processing Systems, Montréal, pp. 1693–1701. PMLR (2015)
8. Yoshida, M., Matsumoto, K., Kita, K.: Distributed representations for words on tables. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 135–146. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57454-7_11
9. Park, G., Lee, H., Kim, H.: Named entity recognition model based on neural networks using parts of speech probability and gazetteer features. Adv. Sci. Lett. **23**(10), 9530–9533 (2017)
10. Feng, M., Xiang, B., Glass, M.R., et al.: Applying deep learning to answer selection: a study and an open task. In: International Conference on Automatic Speech Recognition and Understanding, Scottsdale, AZ, pp. 813–820. IEEE (2016)
11. Santos, C., Tan, M., Xiang, B., et al.: Attentive pooling networks. arXiv preprint arXiv:1602.03609, pp. 1–10 (2016)
12. Salamon, J., Bello, J.P.: Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Sig. Process. Lett. **24**(3), 279–283 (2017)
13. Yang, Y., Wang, F., Zhang, J., Xu, Y., Philip, S.: A topic model for co-occurring normal documents and short texts. World Wide Web **21**(2), 487–513 (2018)
14. Fu, J., Qiu, X., Huang, X.: Convolutional deep neural networks for document-based question answering. In: Lin, C.-Y., Xue, N., Zhao, D., Huang, X., Feng, Y. (eds.) ICCPOL/NLPCC - 2016. LNCS (LNAI), vol. 10102, pp. 790–797. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50496-4_71
15. Tan, M., Santos, C.D., Xiang, B., Zhou, B.: LSTM-based deep learning models for non-factoid answer selection. arXiv preprint arXiv:1511.04108, pp. 1–11 (2015)

Features' Associations in Fuzzy Ensemble Classifiers

Ilef Ben Slima[1(✉)] and Amel Borgi[2]

[1] Faculté des Sciences de Tunis, LIPAH, University of Tunis El Manar, 2092 Tunis, Tunisie
ilef.benslima@fst.rnu.tn
[2] Institut Supérieur d'Informatique, LIPAH, University of Tunis El Manar, 2080 Tunis, Tunisie
Amel.Borgi@insat.rnu.tn

Abstract. In this work, we are interested in ensemble methods for fuzzy rule-based classification systems where the decisions of different classifiers are combined to form the final classification model. We focus on ensemble methods that cluster the set of attributes into subgroups and treat each subgroup separately. This allows decomposing the learning problem into sub-problems of lower complexity and obtaining more intelligible rules as their number and size are smaller. In this paper, we study different methods that allow finding associations between the attributes. In this context, SIFRA is an interesting attributes regrouping method based on association rules concept. We compare SIFRA with some other association methods and show, via a detailed analysis of experimental results, that it is able to find interesting types of associations including linear and non-linear ones. Moreover, it improves the system's accuracy and guarantees a smaller rules number compared to classical FRBCS.

Keywords: Attributes regrouping · Frequent itemsets mining
Associations measures · Correlation analysis · FRBCS · Ensemble methods

1 Introduction

Fuzzy Rule-Based Classification System (FRBCS) is a well-known and widely used tool in supervised machine learning since it provides easily interpretable models using linguistic if-then rules. In these systems, fuzzy rules are automatically generated from numerical data [1, 2]. However, a large number of descriptive attributes can lead to the explosion of the generated rules number. In this context, different works focused on reducing the rules number and decreasing the system's complexity. We mention, for example, the rule selection [3, 4] and the feature selection approaches [5]. Another interesting approach used to reduce the complexity of a FRBCS without excluding any attribute is the attributes regrouping approach. It consists in decomposing the learning problem into sub-problems with lower complexity [6]. In other words, the attributes of the learning problem are regrouped into subgroups; and the attributes of each subgroup are treated separately. This idea has been initially proposed in [6] in a non-fuzzy context and then extended in [7] in the FRBCS context. In these works, the attributes

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 369–377, 2018.
https://doi.org/10.1007/978-3-319-98812-2_33

regrouping method is performed by linear correlation research among the training set elements. The drawback of these methods, such as SIFCO [7], is that they only search for linear correlations between the attributes. However, different other types of dependencies can exist (such as polynomial and curvilinear). Another method of attributes regrouping, called SIFRA, was proposed in [8]. It used the association rules concept, more precisely frequent itemsets mining. In [8], an evaluation of SIFRA in terms of good classification rates has been made in comparison with the SIFCO method [7]. However, no analysis of the types of detected associations has been made. In addition, the results in [8] showed that the two methods gave similar results in almost all the databases. We present in this paper an extended experimentation of SIFRA and we make a detailed analysis of the results in order to show the goodness of SIFRA compared to other fuzzy ensemble methods, such as SIFCO. Thus, we explore in Sect. 2 different measures of associations proposed in the literature and present in Sect. 3 the SIFRA method. A detailed analysis of the type of used data is performed in Sect. 4 where we prove that SIFRA is more suitable for databases which don't have any linear correlations between the attributes. Finally, we analyze in Sect. 5 the detected groups of attributes and the different types of associations found by SIFRA in comparison with SIFCO.

2 Associations Between the Attributes in FRBCS

The attributes regrouping approach is based on ensembles of learning machines where the decisions of different learners are combined to make the final decision. Taking into account the opinions of several experts rather than one single opinion can improve the performance of the overall system [9]. In this work, we focus on ensemble methods where the information to be cluster concerns the attributes. These methods are especially useful for high dimensional databases. Random subspace [10] is one of the first methods proposed to divide the set of attributes into subgroups where the attributes are randomly selected. SIFCO is another ensemble method which uses the "Bravais-Pearson" coefficient to detect the correlated attributes [7, 11]. Other measures of correlations or associations can be used to detect the related attributes such as the Cramer's contingency coefficient which is based on the chi-square statistic or the Symmetrical Uncertainty (SU) factor which measures the interaction between two nominal variables [9]. Another interesting method for searching interactions between the attributes was proposed in [8]. This method, called SIFRA, does not search for linear correlations or any other specific form of dependencies but it rather searches for the subspaces having many regions with high density of data.

3 SIFRA Method

SIFRA is an ensemble method that uses an attributes regrouping approach to construct the related attributes groups and treats each group with a different classifier. Each classifier generates his local rule base using the FRBCS learning method of [1]. Finally, the generated local rule bases are combined to form the global rule base.

The regrouping attributes phase of SIFRA is based on the Association Rules (AR) concept, and precisely on Frequent Itemsets Mining. The SIFRA method consists of three steps: (1) mining frequent itemsets, (2) detecting the associated attributes, (3) selecting the final groups of attributes.

3.1 Mining Frequent Itemsets

Several mining frequent itemsets algorithms have been proposed in the literature such as Apriori [12] and FP-Growth [13]. In [8], Apriori was applied as a well-known and widely used algorithm. It generates frequent itemsets using their support values. We remind that an itemset is frequent if its support is greater than a predefined threshold *minsupp*. Since all the attributes are numerical in a FRBCS, they are split into intervals using a regular discretization, and Apriori is performed on the obtained intervals.

3.2 Detection of the Associated Attributes

To find out if some attributes are associated or not, the associations between their intervals are used. For that, an association grid that highlights the associations between the intervals is defined. In this grid, each axis corresponds to one attribute and spells out its intervals. Figure 1 shows an association grid concerning two attributes X_1 and X_2. When two intervals are associated, the cell of the grid corresponding to their intersection is colored in gray: this cell is called a linked region. A linked region is characterized by the density of data that it contains. The definition of the association degree β takes into account the number of linked regions as well as their densities. A threshold β_{min} is defined and the groups whose degrees β are greater than β_{min} are considered as related.

3.3 Selection of the Final Groups of Attributes

From the last step, all the groups of associated attributes are obtained. These groups are of different sizes and can intersect each other. Besides, the attributes regrouping approach aims to construct the groups in such a way that they form a partition of the original attributes set [8]. For this reason, a selection process, based on the next two criteria, was proposed in order to select the most interesting groups of attributes:

- The higher the degree β is, the stronger the relation between the attributes is.
- The groups with more attributes are preferred as they may improve the accuracy.

4 Performance Evaluation of the SIFRA Method

From the experimental tests of [8], we noticed that the classification results strongly depend on the tested data. Thus, we propose to extend this experimentation to consider more datasets and to study the data types: we intend to distinguish between databases with linear correlation and databases with no linear correlations between the features.

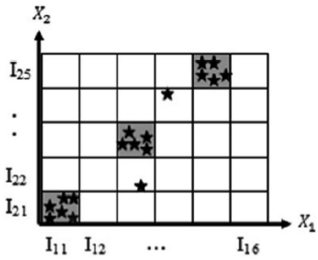


Fig. 1. An example of association grid

For that, we used RStudio[1] software which presents the correlation matrix of a database in a graphical way: the correlation degrees between the attributes are represented by blue circles. The darker and larger the circle is, the more important the correlation is. We chose 8 different databases[2]: Iris (which contains 4 attributes and 150 examples), Wine (13 attributes and 178 examples), Vehicle (18 attributes and 846 examples), Sonar (60 attributes and 208 examples), Glass (9 attributes and 214 examples), Diabetes (8 attributes and 768 examples), Heart-Statlog (13 attributes and 270 examples) and Ecoli (7 attributes and 336 examples). Figure 2 shows some examples of graphical correlation matrices. High linear correlations are detected for the dataset Vehicle (Fig. 2a). However, we do not remark any important correlation for the Diabetes dataset (Fig. 2b). Using these results, we distinguish two groups of data:

- In the first group, Iris, Wine, Vehicle and Sonar have high linear correlations.
- In the second group, the databases Glass, Diabetes, Heart-Statelog and Ecoli represent data with no linear correlations.

We compare SIFRA to other ensemble methods, namely Random Subspace [10] and SIFCO [7]. We should notice that the choice of the method used to find interactions between the attributes has a big impact on the classification task. We consider in this paper the three measures of dependencies presented in Sect. 2: Pearson, Cramer and SU measures. We run the random subspace method using WEKA software[3]. The other methods were run using our own implementation. The two methods SIFCO and SIFRA require the predefinition of some parameters (the correlation threshold, *minsupp*, β_{min}, …). In this paper, we have chosen the values which gave the best results.

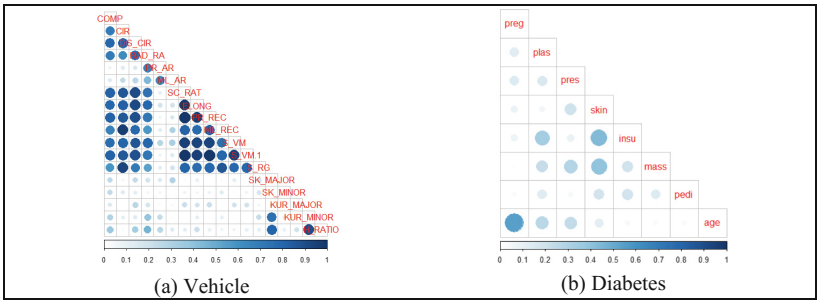
In Table 1, we present the correct classification rates followed by the number of generated rules in brackets. For the random subspace method, WEKA does not provide the rules number; only the classification rates are presented.

From Table 1, SIFRA and SIFCO considerably outperform Random subspace method. In fact, Random subspace randomly generates the attributes groups while SIFCO and SIFRA search for specific interactions between the attributes.

[1] https://www.rstudio.com/products/RStudio/.

[2] https://archive.ics.uci.edu/ml/datasets.html.

[3] https://weka.wikispaces.com/Related+Projects.



(a) Vehicle (b) Diabetes

Fig. 2. Correlation matrices of some databases

Table 1. Comparison of the performance of SIFRA with other ensemble methods

Database	Random subspace	SIFCO (Cramer's coef.)	SIFCO (SU factor)	SIFCO (Pearson's coef.)	SIFRA
Iris	68.66	97.33 (14)	**98.0 (14)**	96.0 (18)	97.33 (14)
Wine	93.25	98.87 (38)	98.87 (38)	98.87 (38)	98.87 (38)
Vehicle	54.96	60.28 (102)	55.43 (62)	60.99 (126)	**67.73 (217)**
Sonar	-	66.34 (81)	66.34 (81)	**70.19 (68)**	66.35 (78)
Glass	52.80	60.75 (21)	60.75 (21)	60.75 (21)	**62.15 (22)**
Diabètes	63.93	71.48 (21)	71.48 (21)	71.48 (21)	**76.30 (36)**
Heart-s	76.66	68.88 (18)	68.88 (18)	70.74 (18)	**77.40 (14)**
Ecoli	48.51	53.57 (16)	55.95 (18)	55.95 (18)	**63.98 (18)**

Regarding the three versions of SIFCO (using different association measures), we notice that SU and Cramer's measures improve the results only in the case of Iris database. However, same or better results are obtained by Pearson's coefficient in all the remaining datasets. As a general conclusion, Bravais-Pearson seems to be better than the Cramer's and SU measures.

Compared to SIFCO, SIFRA almost gives better results except for the Sonar and Iris databases. The same results are obtained by SIFCO and SIFRA in the case of Wine database. For the Vehicle dataset, we observe a significant improvement of the classification rates with SIFRA. However, the rules number obtained by SIFRA is larger. This can be explained by the detection of more subgroups containing more associated attributes (see Table 2). Now, if we consider the second type of used data (data without linear correlations), we remark that SIFRA outperforms SIFCO in all the cases. We also find satisfying results for the rules number: approximately the same rules number as SIFCO for almost all the datasets except Diabetes. These results confirm our hypothesis that SIFRA is more interesting than SIFCO when applied to databases with no linear correlations.

5 Analysis of the Associations Detected by SIFRA and SIFCO

In this section, we make a detailed analysis of the different associations and groups of attributes detected by SIFRA in comparison with SIFCO. We remind that SIFCO uses the linear correlation method and SIFRA uses the AR concept.

5.1 Analysis of the Groups of Attributes

In Table 2, we present the groups of attributes detected by SIFCO and SIFRA. The attributes are presented using numerical symbols in order to simplify the notation (symbol 1 refers to the attribute X_1 in the corresponding database). Table 2 shows that SIFCO and SIFRA provide the same groups of attributes in the case of Wine, that's why the same classification rates are found in Table 1. For the Iris database, the group {3, 4} was detected by the two methods, besides SIFCO associated the attributes 3 and 4 with the attribute 1. In the case of Vehicle, the two methods have detected different groups of attributes. Furthermore, some of the associations were found by both of the two methods, as the associations {9, 11} and {3, 8, 12}. For Sonar database, SIFCO and SIFRA gave totally different groups of attributes.

In the last four databases (with no linear correlations), SIFCO does not find any association between the attributes: each attribute is considered as independent. This is expected since SIFCO searches only for linear correlations.

As conclusion, the experimental results confirm the hypothesis that SIFRA is more interesting than SIFCO when applied to databases with no linear correlations. SIFCO is not able to construct any group of attributes for that type of data. Nevertheless, SIFRA is able to detect different types of associations, including the linear correlations. In the next paragraph, we propose to analyze the shape of associations discovered by SIFRA to better understand what type of associations it searches for.

5.2 Analysis of the Types of Associations

RStudio gives a graphical representation of the data distribution between each pair of attributes. Using these graphics, we can observe the dependencies shapes and we may make an empirical analysis of the types of detected associations. We have analyzed all the groups of attributes obtained by the two methods SIFCO and SIFRA. We present in Fig. 3 some associations discovered by SIFCO only. Figure 4 contains the associations obtained by both SIFCO and SIFRA. Finally, Fig. 5 represents some examples of the associations found by SIFRA only. Each graphic in the figures represents the data distribution according to two attributes.

We observe that most of the associations discovered by SIFCO are linear correlations. Nevertheless, SIFCO has also detected some non-linear correlations as in Figs. 3c and 4c, which is unexpected since SIFCO is supposed to search only for linear correlations. It has been mentioned in [11] that the Pearson's coefficient is not so robust and that it can sometimes reveal non-linear correlations and considers them as linear. We see from Fig. 4 that linear correlations were also detected by SIFRA. Furthermore, SIFRA is able to find other types of associations (see Fig. 5).

Table 2. Comparison of the groups of attributes obtained by SIFRA and SIFCO

Database	Groups of attributes detected by SIFCO	Groups of attributes detected by SIFRA
Iris	{2}, **{1, 3, 4}**	{1}, {2}, **{3, 4}**
Wine	{1}, {2}, {3}, {4}, …, {12}, {13}	{1}, {2}, {3}, {4}, …, {12}, {13}
Vehicle	{1}, **{2, 10, 13}**, **{3, 7, 8, 9, 11, 12}**, {4}, {5}, {6}, {14}, {15}, {16}, {17}, {18}	**{2, 5, 6, 9, 11, 14}**, **{4, 18}**, **{7, 17}**, {13}, **{3, 8, 12}**, {15}, {16}, {1}, {10}
Sonar	{1}, {2}, {3}, …, {14}, **{15, 16}**, **{17, 18}**, {19}, **{20, 21}**, {22}, …, {60}	{1}, **{2, 3, 4, 60}**, {5}, {6}, {7}, {8}, …, {50}, **{51, 52}**, {53}, …, {60}
Glass	{1}, {2}, {3}, … {8}, {9}	**{2, 3, 6, 8}**, **{5, 7, 9}**, **{1, 4}**
Diabètes	{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}	**{4, 5, 8}**, **{1, 2, 6}**, **{3, 7}**
Heart-Statlog	{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}, {11}, {12}, {13}	{1}, **{2, 13}**, **{3, 9}**, {5}, **{4, 7}**, {6}, {8}, {10}, {11}, {12}
Ecoli	{1}, {2}, {3}, {4}, {5}, {6}, {7}	**{1, 3, 4, 5}**, **{2, 7}**, {6}

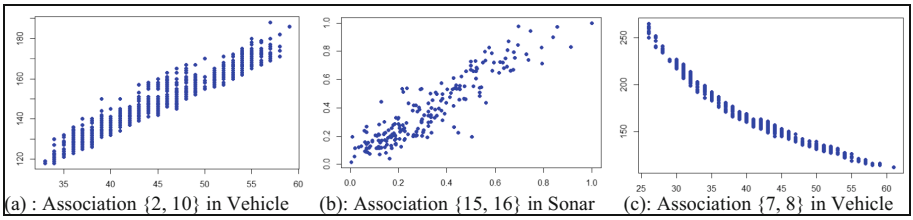


Fig. 3. Associations detected by SIFCO

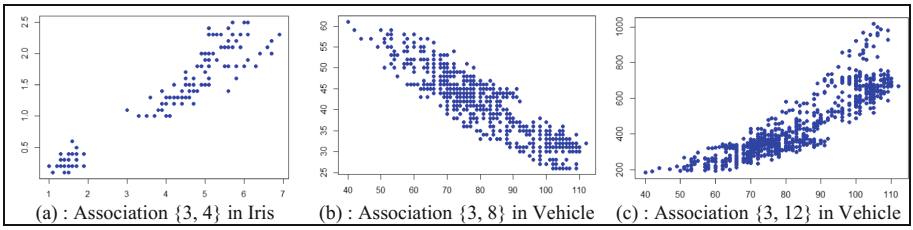


Fig. 4. Associations detected by SIFCO and SIFRA

The graphics (a), (b) and (c) from Fig. 5 represent a high concentration of the data in a region of the pattern space. Other types of non-linear and non-monotonic associations are shown in graphics (d) and (e). Finally, a constant function between two attributes is also considered as an association in SIFRA (see graphics (f)). The experimental results show that SIFRA detects different types of dependencies including the linear and non-linear correlations. In fact, it does not search for a specific form of dependency, but focuses on the concentration of data in some regions of the pattern space.

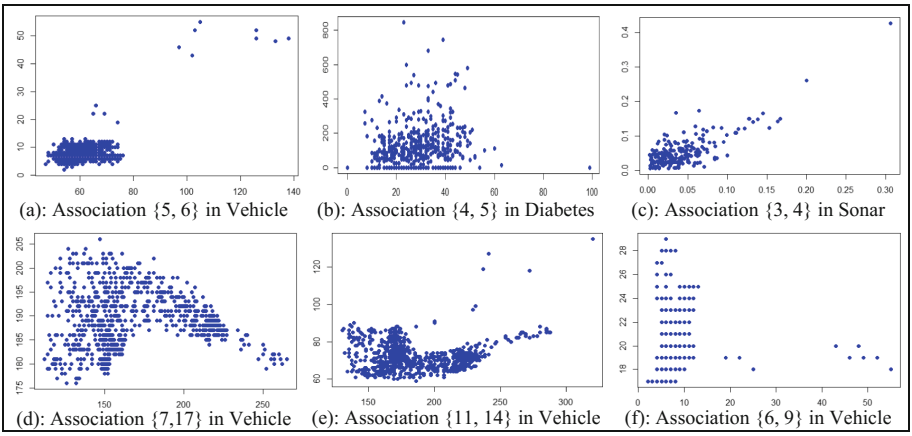


Fig. 5. Associations detected by SIFRA

6 Conclusion

In this paper, we focused on SIFRA, an original method for attributes regrouping based on Association Rules concept, and particularly, on Frequent Itemsets Mining. Experimental tests were performed on different databases and have led to very satisfactory results, especially in the case of data with no linear correlations. SIFRA has the interest of identifying different types of associations between the attributes other than linear correlations. It does not search for a special form of dependency, but it rather detects the concentration of data in some regions of the pattern space. As a perspective, we envisage to study the relationship between the threshold values (*minsupp* and β_{min}) and the data characteristics. It is also interesting to use a method to automatically choose the appropriate values of these thresholds. In addition, a regular discretization was used at the attributes regrouping phase; it would be interesting to use other methods of discretization, such as supervised discretization.

References

1. Ishibuchi, H., Nozaki, K., Tanaka, H.: Distributed representation of fuzzy rules and its application to pattern classification. Fuzzy Set Syst. **52**(1), 21–32 (1992)
2. Dehzangi, O., Zolghadri, M.J., Taheri, S., Fakhrahmad, S.M.: Efficient fuzzy rule generation: a new approach using data mining principles and rule weighting. In: Fuzzy Systems and Knowledge Discovery, FSKD 2007, vol. 2, pp. 134–139 (2007)
3. Rudziński, F.: A multi-objective genetic optimization of interpretability-oriented fuzzy rule-based classifiers. Appl. Soft Comput. **38**, 118–133 (2016)
4. Alcalá, R., Gacto, M.J., Herrera, F., Alcalá-Fdez, J.: A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. Uncertain. Fuzziness Knowl.-Based Syst. **15**, 539–557 (2007)

5. Yu, L., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the 20th International Conference on Machine Learning, pp. 856–863 (2003)
6. Borgi, A., Bazin, J.-M., Akdag, H.: Two methods of linear correlation search for a knowledge based supervised classification. In: Mira, J., del Pobil, A.P., Ali, M. (eds.) IEA/AIE 1998. LNCS, vol. 1415, pp. 696–707. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-64582-9_802
7. Soua, B., Borgi, A., Tagina, M.: An ensemble method for fuzzy rule-based classification systems. Knowl. Inf. Syst. **36**(2), 385–410 (2013)
8. Ben Slima, I., Borgi, A.: Attributes regrouping by association rules in the fuzzy inference systems. Regroupement d'attributs par règles d'association dans les systèmes d'inférence floue. In: EGC 2015, Luxembourg, vol. RNTI-E-28, pp. 317–328 (2015)
9. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Kaufmann, Burlington (2011)
10. Skurichina, M., Duin, R.: Bagging, boosting and the random subspace method for linear classifiers. Pattern Anal. Appl. **5**(2), 121–135 (2002)
11. Saporta, G.: Probabilité, analyse des données et statistique, 2nd edn. Editions Technip, Paris Cedex (2006)
12. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th Very Large Data Bases Conference, VLDB 1994, vol. 1215, pp. 487–499 (1994)
13. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD Record, vol. 29, pp. 1–12 (2000)

Learning Ranking Functions by Genetic Programming Revisited

Ricardo Baeza-Yates[1], Alfredo Cuzzocrea[2,3(✉)], Domenico Crea[4], and Giovanni Lo Bianco[4]

[1] Northeastern University at Silicon Valley, San Jose, CA, USA
rbaeza@acm.org
[2] DIA Department, University of Trieste, Trieste, Italy
alfredo.cuzzocrea@dia.units.it
[3] ICAR-CNR, Rende, Italy
[4] DIMES Department, University of Calabria, Rende, Italy
{dcrea,globianco}@dimes.unical.it

Abstract. We revisit the use of *Genetic Programming* (*GP*) to learn ranking functions in the context of web documents, by adding linking information. Our results show that GP can cope with larger sets of features as well as bigger document collections, obtaining small improvements over the state-of-the-art of GP learned functions applied to web search.

1 Introduction

Searching is still the most common task over the Web. Nevertheless, retrieving *all* the desired information and *only* it, with a high accuracy degree, is still a challenge, even for the most popular web search engines like *Baidu, Bing, Google*, or *Yandex*. From the user's point of view, the key component of a search engine is a ranking algorithm that orders the results by the perceived relevance of documents to the user query.

In *Information Retrieval* (IR), the *Web Ranking Problem* [2] is formally defined as follows. Given a query Q and a collection of web documents D, which maybe HTML pages, textual data or other web data formats, find an *ordering* for the top k most relevant documents from D based on the *relevance* score between each document and the query Q. This relevance score is obtained through a similarity model that can compare documents and/or queries. For example, one of the most popular models for textual documents is the so-called *vector-based model* [2]. The vector-based model is an *algebraic* model that represents documents and queries in terms of vectors where each dimension weight represents a *keyword,* using the cosine distance to measure similarity. One of the most used weight schemes is the so-called *term frequency – inverse document frequency (tf.idf)*. The rationale is that most frequent terms in a document are also the most significant because they tend to be relevant for the document's content. On the contrary, terms that occur in too many different documents are less relevant to discriminate among documents. Hence, the keywords that have high value in both factors are the most discriminative ones. A very popular variant of the *tf.idf* scheme is the *OkapiBM25* ranking function [11], which, due to its good

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 378–386, 2018.
https://doi.org/10.1007/978-3-319-98812-2_34

performance, it is applied to many real-life applications and is commonly used as basic baseline ranking.

In the specific context of web documents, content is semi-structured, hence ranking methods also include functions based on links among HTML pages, such as *PageRank* [5]. Another key component of web search ranking is usage data in the form of clicks. Trying to combine all these variables within the core layer of a meaningful ranking function is not trivial. Hence, the most used technique is called *learning to rank*. In this approach, machine learning is used to learn the ranking with either manually labeled training data or usage data (or both). Although this technique is effective, the actual ranking function is buried in the algorithm itself. An alternative, explored little, that allows learning the ranking function, is to use *Genetic Programming* (*GP*) [3] applied to text documents [10], web documents [1], web advertisement [9], web users' feedbacks [14] or question answering systems [15]. In this paper, we revisit and reengineer this idea, experimentally assessing GP to learn a ranking function in the context of searching web documents adding link features, which implies more features and larger collections.

GP was initially proposed to solve specific search and optimization problems in a wide number of application scenarios. Starting from an input population of *individuals*, a GP algorithm performs a fully-parallel search within the space of solutions. Individuals *(i.e.*, solutions) of a population that seems suitable to solve the problem, given a certain *fitness function*, are selected and combined to generate a new population (or new *generation*). This new population is evaluated again, repeating the process until a *stable state* is reached, *i.e.*, until there are no significant improvements for the best solutions, or until a fixed number of iterations have been performed.

Following previous work, we model individuals as ranking functions and we exploit GP to compose new ranking functions, leading to our CombGenRank approach. In each iteration, ranking functions are evaluated based on their ranking performance against a given web document collection and with respect to queries from an *apriori*-known *training set*. One key difference of our work with respect to previous work, is that in the context of web documents, not only content features are used, but also link features, and can easily be extended to add usage features. Hence, finding a good function is more difficult and is not clear if GP can cope with the increased complexity of the ranking problem. Our results show that GP can cope with more features obtaining small improvements over previous results.

2 Background

In GP, when functions are used, individuals are represented as tress and the three classical genetic operators, i.e. *mutation*, *crossover* and *selection*, are implemented in terms of *tree-like operations (e.g.*, [1, 7, 9, 12]).

Hence, a ranking function in our case is represented as a tree where each node models a genetic operator or a genetic operand. For instance, Fig. 1 depicts the tree representation of a well-known variant of term frequency:

$$f(t,d) = \log_2 \frac{N - (N_t + 0.5)}{N_t + 0.5}$$

where N models the total number of documents in the document collection and N_t is the number of documents in the target collection that contain the term t of the input query.

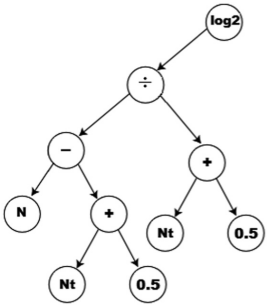


Fig. 1. Example of the representation of a ranking function.

To measure the quality of search results, several performance metrics for ranking functions are available. For instance, *precision-at-document-n* [2] is widely considered as one of the best metrics for the Web, since evaluates the interaction of web user with the first n web pages of the result (typically n is 10), as this is the dominant behavior of web users. However, authors in [6] argue that precision-at-document-n is an *unstable metrics* and demonstrate that MAP [2] is instead a *stable metrics* that is also suitable to more general IR application scenarios. MAP is computed on top of a given set of input queries, by means of the *Average Precision* (PAVG) associated to each query in the input set. Basically, the PAVG associated to a query is given by the sum of the precisions of the query for each *relevant* document retrieved by executing the query, with respect to the total number of documents that are relevant to the query. This definition allows us to assign precision equal to 0 to those documents that are relevant for the query but not retrieved by executing the query itself. Formally, given a query Q and the set of documents D retrieved by executing Q over the target collection, the PAVG of Q, denoted by *PAVG(Q)*, is computed as follows:

$$PAVG(Q) = \frac{\sum_{i=1}^{|D|} \left(r(d_i) * \left(\frac{\sum_{j=1}^{i} r(d_j)}{i} \right) \right)}{Q_{Rel}}$$

where: (*i*) $r(d_i) \in \{0,1\}$ is a parameter equal to 1 if the document d_i is relevant for Q, otherwise it is 0; (*ii*) $|D|$ is the total number of documents that are retrieved by Q; and (*iii*) Q_{rel} is the total number of documents that are relevant for Q.

Given a set of queries W, MAP is defined as follows:

$$MAP = \sum_{i=1}^{W} \frac{PAVG(Q_i)}{|W|} \quad (1)$$

3 CombGenRank: GP Applied to Ranking Web Documents

In CombGenRank, each individual is a ranking function and it is based on the following steps:

1. an initial generation of *M-2* random individuals are generated plus two standard baseline ranking functions, namely OkapiBM25 and PageRank;
2. for each ranking function (i.e., individual), the MAP-based fitness value over the training set is computed (see Eq. 1);
3. the top-k individuals having the best fitness are selected and transferred to the next generation based on the so-called *elitism* concept (*e.g.*, [4]);
4. a new generation of individuals is obtained by randomly applying genetic operators over the current generation;
5. the process above is re-iterated until G generations are obtained and assessed;
6. the best individuals of each generation are assessed against the evaluation set, and their MAP values are stored; and
7. the top-k individuals are finally returned, by both considering ranking performance on the training set and the evaluation set, respectively.

The elitism criterion applied during the selection operation prevents from losing the best individual at the next generation. Without loss of generality, it should be noted that this approach finally allows us to determine the ranking function with the best performance (with respect to the training set) at the last generation.

The CombGenRank algorithm also returns the results on the top-k functions learned during the training phase such that it exposes the best behavior during the evaluation phase. This allows us to avoid selecting functions that over-fit the training set. To this end, we adopted the following formula proposed in [9]:

$$score_i = avg_i - \sigma(avg_i)$$

such that: (*i*) $score_i$ models the score computed for the individual i; (*ii*) avg_i models the average performance of the individual i by considering its performance against the training set and the evaluation set, respectively; and (*iii*) $\sigma(avg_i)$ is the corresponding standard deviation.

Regarding content-oriented parameters, they are the same of the OkapiBM25 ranking functions: (*i*) N: total number of web documents in the collection; (*ii*) N_t: document frequency – it models the total number of documents that contain the term t; (*iii*) T_{ftd}: term's frequency in the document – it models the total number of occurrences of the term in the document; (*iv*) T_{ftq}: term's frequency in the query: – it models the total number of occurrences of the term in the query; and (*v*) T_d: text's length: – it models the total number of terms, including possible duplicates.

Regarding link-oriented parameters, we considered those parameters that typically characterize link quality in web document: (*i*) *In_Links*: the total number of inbound links to the web document; (*ii*) *Out_Links*: the total number of outbound links in the web document; and (*iii*) the *Pagerank value* of the target web page, which is computed according to the original Brin-and-Page's formula in [5], with dumping factor equal to 0.85 and 40 iterations, as argued in [8].

To build ranking functions, both content-oriented and link-oriented parameters are combined by means of the following constant values and arithmetic/functional operators: (*i*) *constant values*: we used 100 constants, denoted by *R1*, *R2*, …, *R100*, such that each constant R_j store a random value selected from the interval [0:100]; (*ii*) arithmetic operators: +, −, *, ÷; and (*iii*) five functional operators: $ln|x|$, $log_2|x|$, *min*, *max*, and $\sqrt{|x|}$. It should be note that the usage of the absolute value in the functional operators is due to the need for avoiding numerical overflows.

When randomly building the first generation, if the number of operands in a certain function is much greater than the number of operators, we would obtain individuals that are "*poorly functional*". Therefore, to alleviate this problem, the probability of selecting an operator is 3 times greater than the probability of selecting an operand or a constant. In addition to this, to avoid obtaining functions that are too deep (hence *syntactically complex*), the probability of selecting an operand decreases as the current number of levels of the tree increases.

Finally, the baseline ranking functions that we use, are especially important during the building of the first generation, to populate it with as many variants of them as possible (of course, the process is later iterated in future generations).

4 Experimental Assessment

In this Section, we provide a preliminary experimental assessment and analysis of CombGenRank in comparison with state-of-the-art techniques, using the well-known WT10G collection. WT10G is a sub-set of the famous WebTREC collection, which has been built to simulate the Web's behavior and that contains more than 1.6 millions web pages, with 311 terms per document in average. For queries, we used TREC 9 topics 451–500 for training and 501–550 for evaluation. Queries with few than five relevant documents were discarded and we also removed stop-words, obtaining 43 and 45 cleaned queries, respectively.

To assess the performance of CombGenRank, we compared its performance against four ranking techniques: (*i*) OkapiBM25 [11], (*ii*) PageRank [3], (*iii*) CCA [1], and (*iv*) *FanGP* [5] (which is an improved version of OkapiBM25). CCA has been implemented via the standard-function blocks reported in [1]. Recall that OkapiBM25 and PageRank are the starting points for our approach and hence we expect to improve upon them. On the other hand, CCA and *FanGP* are the state of the art using the GP approach for web documents.

For the experiments, we use the following probabilities for the genetic operators during the evolutionary process: 0.9 for crossover, 0.05 for mutation, and 0.05 for selection. Regarding the generation of another function, the probability of selecting an operator was set to 0.6, whereas the probability of selecting an operand was set to 0.2.

Finally, we set $M = 100$ individuals for each generation, $k = 10$ for the elite (10%), and $G = 100$ for the number of iterations of our algorithm.

Our experimental methodology comprises 10 runs of the CombGenRank algorithm. For each experimental run, we store the results of the top-10 functions, according to the MAP metrics (see Eq. (1)). Following [12], we split the set of queries to obtain one sub-set of queries to be used during the training phase and another sub-set of queries to be used during the evaluation phase.

Table 1 shows the results of the top-10 functions for the WT10G collection, by reporting the percentage improvements of CombGenRank over (*i*) OkapiBM25 against both, the training set and the evaluation set, and (*ii*) FanGP against the evaluation set.

Figure 2 shows the precision/recall analysis for WT10G of the best ranking function learned by CombGenRank and the comparison approaches. Figure 3 shows the same experimental pattern on the well-known WBR99 collection. These "global" precision and recall associated to the best ranking function have been computed by averaging precisions and recalls computed over each query of the evaluation set.

Table 1. CombGenRank performance for WT10G.

MAP improvements of top-10 functions on WT10G			
#	Training (%)	Evaluation (%)	FanGP (%)
1	+39,35	+15,39	+2.85
2	+39,33	+15,39	+2.85
3	+38,69	+15,26	+2.75
4	+37,06	+15,23	+2.72
5	+36,70	+14,26	+1.75
6	+35,06	+13,78	+1.27
7	+35,06	+13,77	+1.26
8	+35,06	+13,77	+1.26
9	+35,06	+13,77	+1.26
10	+35,08	+13,77	+1.26

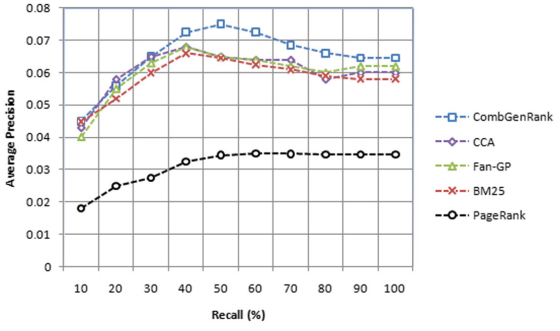


Fig. 2. Precision/Recall analysis for WT10G.

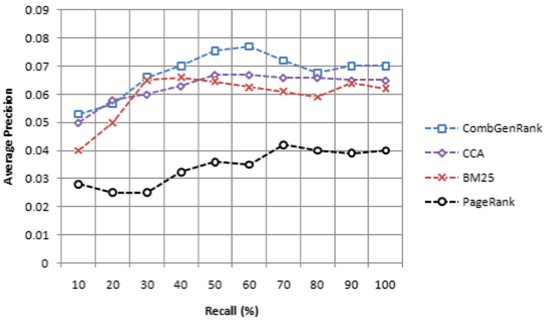


Fig. 3. Precision/Recall analysis for WBR99.

Figure 4 shows the MAP values with respect to the evolution of generations for the running experimental setting over the collection WT10G, while Fig. 5 focuses on the collection WBR99, respectively, for the best ranking function learned by CombGenRank and comparison approaches. For each generation, the average MAP value computed on top of the MAP values retrieved from all the runs with *that* generation has been reported.

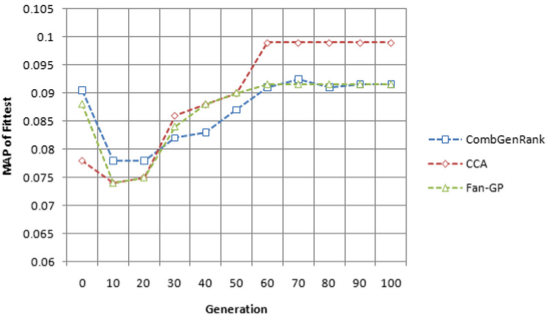


Fig. 4. MAP behavior with respect to Generations over WT10G.

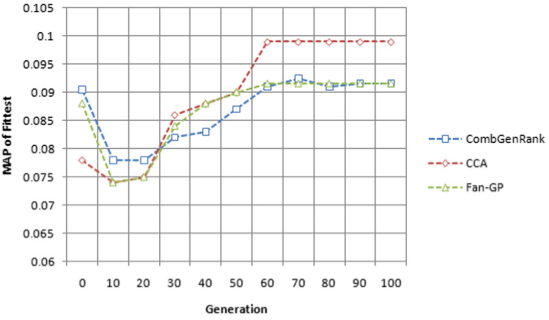


Fig. 5. MAP behavior with respect to Generations over WBR99.

5 Conclusions and Future Work

Our results show that GP might be a viable technique to rank web documents. The main advantage of this technique is that provides interpretability of the results as we can analyze the best ranking functions found. Therefore, this technique can also be used for feature engineering. On the other hand, we need to do further experiments with more relevance features (*e.g.,* include usage data) and larger document collections to assess the real scalability of this technique. In addition, further comparisons with the best learning to rank techniques are needed, studying the sensitivity of all the parameters of our approach as well as including other evaluation measures such as *nDCG* [2]. On a larger vision, future work will focus the attention on extending our technique to make it compliant with novel requirements dictated by emerging *big data trends (e.g.*, [13]).

From the analysis of results, it immediately follows the poor performance of PageRank, as also demonstrated in [8]. This because a function that makes use of link-oriented parameters only cannot take advantages from the content information, as it happened for the successful OkapiBM25. Thanks to the proposed CombGenRank approach, instead, which combines content-oriented and link-oriented parameters, we observed significant performance improvements, with an average value equal to +35.00% over OkapiBM25 against the training set, an average value equal to +15.00% over OkapiBM25 against the evaluation set, and an average value equal to +2.85% over FanGP against the evaluation set, respectively.

Learned ranking functions expose a heterogeneous form, as they contain a full combination of all content-oriented and link-oriented parameters, still confirming that both classes of parameters are necessary in order to obtain good performance.

References

1. Almeida, H.M., Gonçalves, M.A., Cristo, M., Calado, P.: A combined component approach for finding collection-adapted ranking functions based on genetic programming. In: Proceedings of the 30th ACM SIGIR, pp. 399–406 (2007)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval: The Concepts and Technology Behind Search Engines, 2nd edn. Addison-Wesley, Boston (2011)
3. Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P.: Genetic Programming: An Introduction – On the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, Burlington (1998)
4. Bazen, S., Moyes, P.: Elitism and stochastic dominance. Soc. Choice Welfare **39**(1), 207–251 (2012)
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Proceedings of the 7th International Conference on World Wide Web 7, pp. 107–117 (1998)
6. Buckley, C., Voorhees, E.M.: Evaluating evaluation measure stability. In: Proceedings of the 23th ACM SIGIR, pp. 33–40 (2000)
7. Fan, W., Pathak, P., Wu, H.: The effects of fitness functions on genetic programming-based ranking discovery for web search. JASIST **55**, 628–636 (2004)
8. Gevrey, J., Ruger, S.M.: Link based approaches for text retrieval. In: Proceedings of the 10th TREC (2001)

9. Lacerda, A., Cristo, M., Gonçalves, M.A., Fan, W., Ziviani, N., Ribeiro-Neto, N.: Learning to advertise. In: Proceedings of the 29th ACM SIGIR, pp. 549–556 (2006)
10. Oren, N.: Reexamining TF-IDF based information retrieval with genetic programming. In: Proceedings of SAICSIT, pp. 224–234 (2002)
11. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retrieval **3**(4), 333–389 (2009)
12. Trotman, A.: Learning to rank. Inf. Retrieval J. **8**, 359–381 (2005)
13. Cuzzocrea, A., Saccà, D., Ullman, J.D.: Big data: a research agenda. In: Proceedings of IDEAS, pp. 198–203 (2013)
14. Keyhanipour, A.H., Moshiri, B., Oroumchian, F., Rahgozar, M., Badie, K.: Learning to rank: new approach with the layered multi-population genetic programming on click-through features. Genet. Program Evolvable Mach. **17**(3), 203–230 (2016)
15. Khodadi, I., Abadeh, M.S.: Genetic programming-based feature learning for question answering. Inf. Process. Manag. **52**(2), 340–357 (2016)

A Comparative Study of Synthetic Dataset Generation Techniques

Ashish Dandekar(✉), Remmy A. M. Zen, and Stéphane Bressan

National University of Singapore, Singapore, Singapore
{ashishdandekar,remmy}@u.nus.edu, steph@nus.edu.sg

Abstract. Unrestricted availability of the datasets is important for the researchers to evaluate their strategies to solve the research problems. While publicly releasing the datasets, it is equally important to protect the privacy of the respective data owners. Synthetic datasets that preserve the utility while protecting the privacy of the data owners stands as a midway.

There are two ways to synthetically generate the data. Firstly, one can generate a fully synthetic dataset by subsampling it from a synthetically generated population. This technique is known as fully synthetic dataset generation. Secondly, one can generate a partially synthetic dataset by synthesizing the values of sensitive attributes. This technique is known as partially synthetic dataset generation. The datasets generated by these two techniques vary in their utilities as well as in their risks of disclosure.

We perform a comparative study of these techniques with the use of different dataset synthesisers such as linear regression, decision tree, random forest and neural network. We evaluate the effectiveness of these techniques towards the amounts of utility that they preserve and the risks of disclosure that they suffer. We find decision tree to be an efficient and a competitively effective dataset synthesiser.

Keywords: Synthetic datasets · Risk of disclosure · Privacy · Utility

1 Introduction

On one hand, the philosophy of open data dictates that if the valuable datasets are made publicly available, the problems can be crowdsourced in the expectation to obtain the best possible solution. On the other hand, business organizations have their concerns regarding the public release of the datasets which may lead to the breach of private and sensitive information of stakeholders. In order to mitigate the risk of a confidentiality breach, agencies employ different techniques such as reordering or recoding of sensitive variables, shuffling values among different records. In spite of these efforts by agencies, we have examples of confidentiality breaches [11,20] in anonymised datasets.

Fully synthetic datasets proposed by Rubin [18] and partially synthetic datasets proposed by Little [9] bridge the gap between utility and privacy. They

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 387–395, 2018.
https://doi.org/10.1007/978-3-319-98812-2_35

use multiple imputation, a technique used for repopulating the missing values in a dataset, to generate synthetic records which preserve relationships in the population. Following up on these works of multiple imputation, Reiter et al. [1,7,14,16] use different machine learning tools to generate synthetic datasets. These works treat values of synthetically generated attributes as missing values that are generated using models such as Decision Trees, Random Forest, Support Vector Machine, etc.

In this work, we comparatively evaluate fully synthetic dataset generation and partially synthetic dataset generation using different dataset synthesisers: namely linear regression, decision tree, random forest and neural network. We comparatively evaluate effectiveness, as utility preservation and risk of disclosure, and efficiency of the synthetic dataset generation techniques.

Given the tradeoff between the efficiency and effectiveness, we observe that decision trees are not only efficient but also competitively effective compared to other dataset synthesisers.

The extended version of this paper is available at [2].

2 Related Work

Synthetic dataset generation work stems from the early works of data imputation to fill in the missing values in the surveys [17]. In [18], Rubin proposes a procedure to generate fully synthetic dataset that uses multiple imputation technique to synthetically generates values for a set of attributes for all datapoints in the dataset. Although it is advantageous to synthetically generate values for all datapoints, it is not always a necessity. Partially synthetic datasets, proposed by Little [9], are generated by synthetically generating the values of the attributes that are sensitive to public disclosure. Various dataset synthesisers such as decision tree [1,3,16] have been used to generate fully and partially synthetic datasets.

Drechsler et al. [7] have performed an empirical comparative study between different dataset synthesisers. Comparison between fully and partially synthetic datasets can be found in [5]. Recently, Nowok et al. [12] have created an R package, *synthpop*, which provides basic functionalities to generate synthetic datasets and perform statistical evaluation.

The effectiveness of the synthetic dataset lies in the amount of utility it retains from the original dataset. Most of the works [1,7,14,16] use statistical methods of estimation for the evaluation of utility. They use estimators of mean and variance to calculate confidence intervals. Regression analyses are used to test whether the relationships among different variables are preserved. Aside from these analysis specific measures, Woo et al. [21] and Karr et al. [8] have proposed global measures such as Kullback-Leibler (KL) divergence, extension of propensity score and cluster analysis measure.

One of the prime motivations behind publicly releasing synthetic dataset instead of original datasets is to maintain the privacy of the data owners. In [15], Reiter introduces formalism to calculate risk of disclosure in synthetically generated datasets using multiple imputation. The same formalism has

been used in [7,16] to evaluate the risk of disclosure. For further details, readers are requested to refer to [4].

In this work, we comparatively evaluate efficiency and effectiveness of fully and partially synthetic dataset generation techniques using different dataset synthesisers including neural network.

3 Synthetic Dataset Generation Using Multiple Imputation

Multiple Imputation. Consider a dataset of size n sampled from a population of size N. Let Y_{nobs} denote subset of attributes in the dataset whose values are either missing for some datapoints or sensitive towards the public disclosure. Rubin [17] proposes to synthetically generate values for Y_{nobs} given the knowledge of rest of the attributes in the dataset, say Y_{obs}.

Let $\mathcal{M}$ be a dataset synthesiser that generates values for an attribute Y_i given the information about rest of the attributes, denoted as Y_{-i}. With the help of $\mathcal{M}$, an imputer independently synthesises values of Y_{nobs} m times and releases m synthetic datasets $\mathcal{D} = \{\mathcal{D}^1, \mathcal{D}^2, \ldots, \mathcal{D}^m\}$.

In order to synthesise multiple sensitive attributes, we follow the procedure presented in [3]. It defines an order on the attributes that are to be synthesised. Values of the first attribute are synthesised by training the dataset synthesiser on the original dataset. For any later attributes, the dataset synthesiser is trained on the dataset with synthetic values of the attributes preceding it. Interested readers can refer to [16] for a detailed discussion on choosing the order of synthesis.

The reason behind releasing m different datasets and combining estimators on each dataset is two folds. Firstly, there is randomness in the dataset due to sampling from the population. Secondly, there is randomness in the dataset due to imputed values. In order two capture these variabilities, framework of multiple imputation proposes the release of m datasets.

Fully Synthetic Dataset Generation. Consider a dataset of size n sampled from a population of size N. Suppose that an imputer knows the values of a set of variables X for the entire population and values for rest of the variables, Y, only for a selected small sample. Let Y_{inc} and Y_{exc} denote values of variables which are included in the sample and excluded from the sample respectively. The imputer synthetically generates values of Y_{exc} using a dataset synthesizer $\mathcal{M}$ trained on Y_{inc} and X. This synthesis is equivalent to performing multiple imputation with Y_{exc} as Y_{nobs} and Y_{inc} as Y_{obs}. Publicly released datasets, $\mathcal{D}$, comprise of m samples selected synthetically generated population. In order to statistically estimate an attribute Q, we use the estimators of mean and variance presented in [14].

Theoretically, fully synthetic datasets provides 100% guarantee against the disclosure of value of sensitive attribute [18]. Since $n << N$, it is less probable to have record from the original sample in the final dataset. Final datasets are sampled from synthetic population datasets in which $N - n$ records are synthetically generated.

Partially Synthetic Dataset Generation. Let S be a dataset of size n sampled from a population of size N. In order to protect the sensitive information, an imputer decides to alter values of a set of attributes, Y, for a subset of datapoints in S. Let Z be a binary vector of size n. Z_i takes value one if Y values of the i^{th} datapoint are to be synthetically generated and Z_i takes value zero if values of Y attributes are not be altered.

Let $Y_{syn} = \{Y_i | \forall i, Z_i = 1\}$ and $Y_{org} = \{Y_i | \forall i Z_i = 0\}$. We generate m partially synthetic datasets by multiple imputation. In this case, Y_{syn} are the datapoints, with missing values, that we synthetically generate by training a dataset synthesiser on the available data, i.e Y_{org}. This synthesis is equivalent to performing multiple imputation with Y_{syn} as Y_{nobs} and Y_{org} as Y_{obs}. Publicly released datasets, $\mathcal{D}$, comprise of m datasets sampled from the population wherein values of attributes in Y are synthetically generated, as specified by Z, for each of the dataset. In order to statistically estimate an attribute Q, we use the estimators of mean and variance presented in [13].

Dataset Synthesisers. Now, we discuss different dataset synthesisers namely linear regression, decision tree, random forest and neural network.

We use linear regression [10] as a baseline dataset synthesiser. In order to synthesise every attribute Y_i in a dataset, we learn the parameters of the regression model using the dataset with attributes in Y_{-i}. We generate values for Y_i by sampling from a Gaussian distribution with a constant variance and the mean as determined parameters of regression.

We adopt the technique, proposed by Reiter [16], that uses classification and regression tree [10] to generate partially synthetic datasets. The procedure starts with building a decision tree using the values of the attributes that are available in the dataset Y_{-i}. In order to synthesise the value of an attribute Y_i for a datapoint j, we trace down the tree using the known attributes of j until we reach the leaf node. Let L_j be the set of values of Y_i in the leaf node. For a categorical attribute Y_i, Reiter proposes Bayesian bootstrap sampling to choose m different values. For a continuous attribute Y_i, we fit a kernel density estimator over the values in L_j and sample m values from the estimate.

We adopt the technique, proposed by Caiola et al. [1], that uses random forest [10] to generate partially synthetic datasets. In order to synthesise values for a certain attribute Y_i, they train a fixed number decision trees on random samples of training dataset Y_{-i}. For a categorical attribute, the collection of results from constituent decision tree forms a multinomial distribution. m values are sampled from this distribution as the synthetic values for Y_i. For a continuous attribute, they propose use of a kernel density estimator over the results from decision trees and sample values from the estimator.

We use neural network [10] that learns an abstract function mapping an input to the corresponding output as a data synthesiser. If we consider K-class classification problem, the output layer of a neural network comprises of K nodes, with each node representing the probability of the respective class being the output of the model. We treat every attribute as a categorical variable. In

order to synthesise value of an attribute Y_i, we train a neural network using features in Y_{-i}. We sample a value for attribute Y_i using the output layer as a multinomial distribution.

4 Empirical Evaluation

4.1 Dataset and Experimental Setup

We conduct experiments on a microdata sample of US Census in 2000 provided by IPUMS International [19]. Following the approach presented in [7] to consider the records of the heads of households, we consider the records of 316,276 heads of households as the population.

All programs are run on Linux machine with quad core 2.40 GHz Intel® Core i7™processor with 8 GB memory. The machine is equipped with two Nvidia GTX 1080 GPUs. Python® 2.7.6 is used as the scripting language.

4.2 Metrics for Evaluation

The utility of generated dataset needs to be evaluated at two different levels. Firstly, we need to evaluate differences between the distribution of values of original attribute and synthetically generated attributes. Secondly, we need to evaluate the difference between the quality of a statistical estimator for an attribute on synthetic dataset and original data.

In order to evaluate the first level utility, we calculate the similarity between the overall distribution of values of an attribute by calculating normalised KL-divergence between the distribution of values of the attribute in population and the distribution of the same attribute in synthetically generated dataset. In order to evaluate the second level utility, we use the overlap [8] between 95% confidence intervals of an statistical estimator that are obtained using original dataset and synthetically generated dataset. If the intervals are similar to each other, synthetic dataset generation procedure preserves the utility. Therefore, maximum extent of overlap, which is 1, implies preservation of the utility.

We follow the procedure in Reiter [6,14] to estimate the risk of disclosure in the synthetically generated dataset. We assume that the intruder has complete information about an auxiliary variable, say region of birth, which is not a sensitive variable. Let $\mathbf{t}$ be a vector of information possessed by an intruder. For every datapoint j in the dataset, the intruder calculates the probability of the datapoint j being the record of interest.

The intruder selects datapoints with maximum probability value. This process is repeated for every target datapoint in $\mathbf{t}$. In order to evaluate the risk of disclosure, we calculate *true match rate* (True MR) and *false match rate* (False MR) as defined in [6,14]. Smaller the true match rate, better is the performance of a dataset synthesiser.

4.3 Evaluation

The process starts by drawing 1% sample from the population, which we treat as the original dataset. We synthetically generate values for two attributes: income and age, in the same order. We generate five synthetic datasets for each original dataset. We repeat this procedure for 500 original datasets and mean of various metrics over 500 iterations is reported. In order to generate partially synthetic datasets, we need to define the cutoffs for the values of attribute that determine quantify the sensitivity of the attribute towards disclosure. We consider datapoints that have more than 70,000$ income value and less than 26 age value to be the ones with sensitive information.

Utility evaluation results for the *age* attribute for partially synthetic datasets and fully synthetic datasets are presented in Tables 1 and 2 respectively. We observe that although two techniques show comparable values of synthetic means, the technique of partially synthetic dataset generation shows greater extent of the overlap. Partially synthetic dataset generation does not replace all values of the attributes in the sample. Therefore, we observe higher overlap for partially synthetic datasets. We also observe a large deviation in the sample mean of from its original mean in case of linear regression. Linear regression in the absence of any regularization suffers from overfitting [10]. Due to the order of synthesis, linear regression model is fit on the synthetically generated values of *income* while synthesising value for *age*. Thus, it overfits the synthetic data and fails to capture exact distribution of values in the original dataset. Decision tree and other models are not prone to overfitting the training dataset and hence do not show such a degradation in utility. We also conduct similar evaluation for the *income* attribute that we present in the long version of this paper [2].

Table 1. Evaluation of utility for partially synthetic datasets generated using different dataset synthesisers.

Dataset synthesisers	Original sample mean	Partially synthetic dataset		
		Synthetic mean	Overlap	Norm KL div.
Linear regression	49.83	24.69	0.50	0.55
Decision tree	49.83	49.83	0.90	0.56
Random forest	49.82	49.74	0.95	0.56
Neural network	49.87	49.78	0.90	0.56

In order to evaluate the risk of disclosure, we require a scenario. We assume that an intruder is interested in people who are born in US and have income more than 250,000$. All these people are the targets of the intruder. Intruder tries to match every single target with the records in the released datasets. We consider two records perfectly match if the people representing the records are born in US, they have income more than 250,000$ and the age of the person in dataset in within the tolerance of 2 compared to target person.

Table 2. Evaluation of utility for fully synthetic datasets generate using different dataset synthesisers.

Dataset synthesisers	Original sample mean	Fully synthetic dataset		
		Synthetic mean	Overlap	Norm KL div.
Linear regression	49.83	−192.21	0.50	0.56
Decision tree	49.83	49.83	0.56	0.56
Random forest	49.82	46.25	0.68	0.57
Neural network	49.76	54.32	0.75	0.99

Two cases arise in the evaluation. For a given target, the intruder may or may not know if the target person is included in the released sample. We observe that, in the case when the intruder does not have certainty about inclusion of target in the sample, risk of disclosure is the least. In most of the cases, the targets might not be present in the released sample which leads to true match rate of 0. Observing the results for the case when a target is present in the sample, we see that neural network comparatively offer better performance than rest of the dataset synthesisers (Table 3).

Table 3. Evaluation of risk of disclosure for different dataset synthesisers

Dataset synthesiser	Target is in the sample		Target may be in the sample	
	True MR	False MR	True MR	False MR
Linear regression	0.06	0.82	0.00	0.00
Decision tree	0.18	0.68	0.00	0.99
Random forest	0.35	0.50	0.00	0.99
Neural network	0.03	0.92	0.00	0.99

We present the results of comparative efficiency of both these techniques using different dataset synthesisers in Table 4. We observe that the neural network achieve the low risk of disclosure at the cost of a higher running time than the time taken by linear regression or decision trees.

Table 4. Efficiency: each cell shows the running time required, in seconds, to generate five synthetic datasets.

Dataset synthesiser	Partially synthetic dataset generation	Fully synthetic dataset generation
Linear regression	0.040	0.068
Decision tree	0.048	0.533
Random forest	3.350	103.543
Neural network	0.510	55.26

5 Conclusion and Future Works

In this work, we comparatively evaluate fully and partially synthetic dataset generation techniques using different dataset synthesisers, namely linear regression, decision tree, random forest and neural network. We comparatively evaluate effectiveness, in terms of utility preservation and risk of disclosure, and efficiency of these techniques. The analysis shows that decision trees stand as a good dataset synthesiser given its high effectiveness compared to other data synthesisers. This observation agrees with the result in [7].

We use a well-structured dataset in this work. Many real-world datasets do not have a well defined structure. For instance, the social network datasets or the datasets generated from the readings collected by sensors. As a future work, we want to explore how synthetic dataset generation techniques can be adopted for such non-structured or semi-structured datasets.

Acknowledgement. This research is supported by the National Research Foundation, Prime Ministers Office, Singapore, under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd.

References

1. Caiola, G., Reiter, J.P.: Random forests for generating partially synthetic, categorical data. Trans. Data Priv. **3**(1), 27–42 (2010)
2. Dandekar, A., Zen, R.A., Bressan, S.: A comparative study of synthetic dataset generation techniques. Technical report TRA6/18, National University of Singapore, June 2018. https://dl.comp.nus.edu.sg/handle/1900.100/7050
3. Drechsler, J.: Using support vector machines for generating synthetic datasets. In: Domingo-Ferrer, J., Magkos, E. (eds.) PSD 2010. LNCS, vol. 6344, pp. 148–161. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15838-4_14
4. Drechsler, J.: Synthetic Datasets for Statistical Disclosure Control: Theory and Implementation, vol. 201. Springer, Heidelberg (2011). https://doi.org/10.1007/978-1-4614-0326-5
5. Drechsler, J., Bender, S., Rässler, S.: Comparing fully and partially synthetic datasets for statistical disclosure control in the german iab establishment panel. Trans. Data Priv. **1**(3), 105–130 (2008)
6. Drechsler, J., Reiter, J.P.: Accounting for intruder uncertainty due to sampling when estimating identification disclosure risks in partially synthetic data. In: Domingo-Ferrer, J., Saygın, Y. (eds.) PSD 2008. LNCS, vol. 5262, pp. 227–238. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87471-3_19
7. Drechsler, J., Reiter, J.P.: An empirical evaluation of easily implemented, nonparametric methods for generating synthetic datasets. Comput. Stat. Data Anal. **55**(12), 3232–3243 (2011)
8. Karr, A.F., Kohnen, C.N., Oganian, A., Reiter, J.P., Sanil, A.P.: A framework for evaluating the utility of data altered to protect confidentiality. Am. Stat. **60**(3), 224–232 (2006)
9. Little, R.J.: Statistical analysis of masked data. J. Off. Stat. **9**(2), 407 (1993)
10. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge (2012)

11. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: 2008 IEEE Symposium on Security and Privacy. SP 2008, pp. 111–125. IEEE (2008)
12. Nowok, B., Raab, G., Dibben, C.: synthpop: bespoke creation of synthetic data in R. J. Stat. Softw. **74**(11), 1–26 (2016)
13. Raghunathan, T.E., Reiter, J.P., Rubin, D.B.: Multiple imputation for statistical disclosure limitation. J. Off. Stat. **19**(1), 1 (2003)
14. Reiter, J.P.: Inference for partially synthetic, public use microdata sets. Surv. Methodol. **29**(2), 181–188 (2003)
15. Reiter, J.P.: Estimating risks of identification disclosure in microdata. J. Am. Stat. Assoc. **100**(472), 1103–1112 (2005)
16. Reiter, J.P.: Using cart to generate partially synthetic public use microdata. J. Off. Stat. **21**(3), 441 (2005)
17. Rubin, D.B.: Basic ideas of multiple imputation for nonresponse. Surv. Methodol. **12**(1), 37–47 (1986)
18. Rubin, D.B.: Discussion statistical disclosure limitation. J. Off. Stat. **9**(2), 461 (1993)
19. Ruggles, S., Genadek, K., Goeken, R., Grover, J., Sobek, M.: Integrated public use microdata series: Version 6.0 [dataset] (2015). https://doi.org/10.18128/D010.V6.0
20. Sweeney, L.: Computational disclosure control for medical microdata: the datafly system. In: Record Linkage Techniques 1997: Proceedings of an International Workshop and Exposition, pp. 442–453 (1997)
21. Woo, M.J., Reiter, J.P., Oganian, A., Karr, A.F.: Global measures of data utility for microdata masked for disclosure limitation. J. Priv. Confid. **1**(1), 7 (2009)

Emerging Applications

The Impact of Rainfall and Temperature on Peak and Off-Peak Urban Traffic

Aniekan Essien[1(✉)], Ilias Petrounias[1], Pedro Sampaio[1], and Sandra Sampaio[2]

[1] Alliance Manchester Business School, University of Manchester, Oxford Road, Manchester M13 9PL, UK
{aniekan.essien, ilias.petrounias, pedro.sampaio}@manchester.ac.uk
[2] School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK
sandra.sampaio@manchester.ac.uk

Abstract. This paper focuses on quantifying the effect of rainfall and temperature intensities on urban traffic characteristics at peak and off-peak periods respectively using traffic data from Greater Manchester, UK, as case study. Three broader issues are addressed: (1) the impact of rainfall on urban traffic; (2) the impact of rainfall intensity on traffic flow parameters at peak and off-peak periods respectively; (3) the impact of atmospheric temperature level on peak and off-peak urban traffic. Our contribution arises both from the combination of factors included in the study as well as distinct analyses of peak and off-peak traffic data. This is the first study undertaken in a real urban environment with reduced operating speed (30 mph), and can provide urban traffic policymakers with crucial information that can be used to modify or develop traffic planning decisions in order to maximize traffic network utilization.

Keywords: Traffic analytics · Traffic data analysis · Data science

1 Introduction

Over the years, it has been identified that harsh weather can considerably affect traffic flow parameters by influencing driving behaviour, travel demand, travel mode, safety, as well as traffic flow characteristics [1–5]. In terms of traffic operation, rain conditions reduce traffic capacity and operating speeds, thereby increasing congestion and road network productivity loss. The incorporation of weather-related data for traffic management enables a clear understanding of the dynamics of traffic flow modeling, especially in geographical locations with the tendency of severe weather conditions. Despite the vastness of studies investigating whether weather affects traffic, there is still a dire need to quantify the impact of weather conditions on traffic operation within urban road networks. The limitation in understanding the direct impact on traffic by weather conditions minimizes the potential for transportation policymakers to capitalize on additional intelligence provided by weather-related data sources in order to develop improved traffic management strategies. This leaves the presence of a number of gaps or opportunities for new and exciting research within this area.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 399–407, 2018.
https://doi.org/10.1007/978-3-319-98812-2_36

One of the main limitations addressed by this study is the fact that only few studies have investigated the impact of weather conditions on urban traffic. This is possibly because urban links are shorter in length, usually have lower speed limits, and have more interrupted traffic [6]. Secondly, peak and off-peak periods should be considered during these weather-traffic analyses. Policymakers can benefit more from a deeper understanding of the effects of weather conditions on peak and off-peak periods, leading to better planning decisions or traffic control modifications that will be applied to effectively control traffic congestion. This research differs from existing studies by focusing on weather impact on urban traffic rather than freeway/motorway/highway traffic.

The remainder of this paper is organised as follows. Section 2 presents related studies, while the Methodology and Data are presented in Sect. 3. The results are presented and discussed in Sect. 4. The final section presents the conclusion and future research opportunities.

2 Related Work

Many studies have attempted to investigate the relationship between weather and traffic. For instance, Jones and Goolsby [7] reported a 14% reduction in operating speed during rainfall. Smith and Byrne [8] observed that light rain decreased capacity by 4 to 10%, while heavy rain caused a reduction in the range of 25 to 30%. Ibrahim and Hall [9] performed a regression analysis and found out that light rain caused a reduction of about 1.1 mph in operating speeds during free-flowing conditions, 4 to 8 mph in congested conditions, and 8 to 10 mph in capacity conditions. Mahmassani and Dong [10] showed that adverse weather can affect traffic by increasing demand, as well as shifting peak-hour demand. Similarly, Wang and Yamamoto [11] observed a reduction in operating speeds to about 6 to 8% during heavy rainfall and that the magnitude of the weather impact was dependent on road network characteristics, such as number of lanes and size of road. Other studies claim that rainfall intensity is what negatively affects the traffic speed [2, 6, 8, 12].

A step in the right direction would be the analysis of the impacts of weather on traffic within an urban setting, as opposed to the vast majority of studies that consider congested/uncongested freeway traffic [3, 9, 13–16]. In addition to this, identifying how the traffic is impacted at peak and off-peak times provides valuable additional information to policymakers, for developing improved traffic management policies.

3 Methodology and Data

The traffic data used within this study was provided by the Transport for Greater Manchester (TfGM) and comprised observations of average speed, flow, density, and travel time. The data was collected using a combination of Inductive Loop Devices (ILD) and Bluetooth sensors on an urban arterial road (Chester Road - A56) in Stretford,

Greater Manchester, UK, as depicted in Fig. 1. This represents an ideal characteristic of serving as a conduit from a residential area to the city centre. Landmark locations around are the Old Trafford Stadium (Manchester United) in addition to other leisure points like shopping malls, clubs, restaurants, etc. The study period spans from 1 January 2014 to 31 December 2014. The study area has a speed limit of 30 mph.

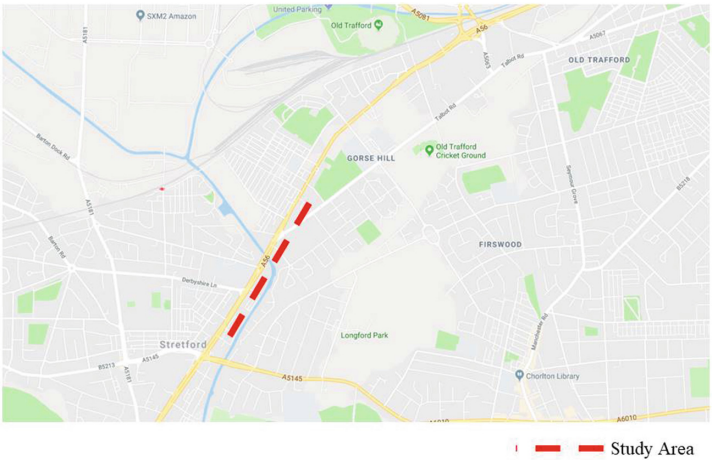


Fig. 1. Map of study area (Source: Google Maps)

The weather data obtained during the study period comprised aggregated hourly observations of temperature (°C) and precipitation (*mm*). The rainfall was categorized as *None, Light* (<0.5 mm), *Moderate* (0.5 mm $< r <$ 4.0 mm), and *Heavy* (r $>$ 4.0 mm) as stated in the SYNOP FM-12 weather equipment. The weather data was obtained from the Centre for Atmospheric Studies (CAS), University of Manchester. The weather stations are located within a 3-mile radius of the study area. Holidays and weekends were excluded from the analysis, as these would have presented significantly differing traffic patterns to the daily peak and off-peak patterns used within this study. The workdays were further sub-categorized into peak and off-peak hours. The peak hours were identified by performing descriptive statistics on the data, which revealed the peak hours (excluding outliers) to be 08:00, 09:00, 10:00, 17:00, 18:00 and 19:00.

4 Results

Speed-density-volume (V-K-Q) plots for peak and off-peak hours were produced, which are presented in Fig. 2. This was compared to existing traffic stream models in the literature [17] and agreed with the Greenshields traffic stream model. Table 1 presents the descriptive statistics for the traffic dataset. All the tables also present the Standard Deviation (SD), Standard Error (SE), Degree of Freedom (*df*) and Mean Square (MS) for the respective groups. As shown in Table 1, all p-values are statistically significant at the 0.05 level, implying that the null hypothesis cannot be rejected.

Also, the average speed at off-peak periods is higher than that of the peak periods, which makes sense and shows the level of traffic congestion during peak hours.

Summarily, there is a 35% reduction in average speed during peak periods, an increase of 157.5% and 154% for volume and density respectively. This represents a significant increase in traffic congestion during peak periods.

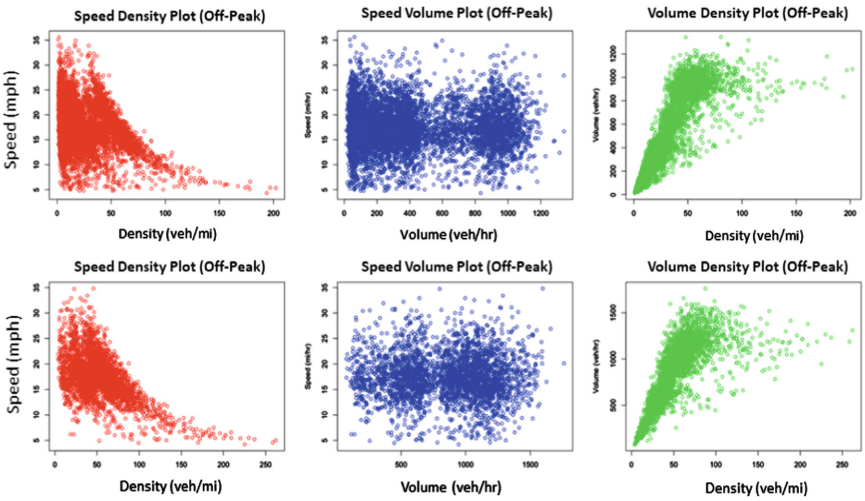


Fig. 2. V-K-Q relationships for peak and off-peak periods

4.1 Precipitation

Table 2 displays the descriptive statistics and single factor ANOVA for the groups of rainfall intensities during peak periods. As can be seen from the table, there is a significant difference $(p < 0.05)$ between all the rainfall intensities and traffic flow characteristics (i.e. speed, volume, and density). What stands out in the table is that heavy rainfall negatively impacted average operating speeds by 9.74%, equivalent to a 2–3 mph average speed reduction. More specifically, average speeds were reduced by 2.61% and 9.74% during moderate and heavy rainfall respectively.

4.2 Temperature

The temperature observations were classified as Freezing (below 5 °C), Cold (between 5 °C and 10 °C), Normal (from 10 to 15 °C), Warm (between 15 °C and 20 °C), and Hot (above 20 °C). Tables 3 and 4 present the descriptive statistics for the respective groups, at off-peak and peak periods respectively. In Table 4, there is a clear trend of increase in volume with temperature at peak periods. However, average speed is reduced by 36.5% at hot temperatures, which translates to a speed reduction of 5–6 mph. Hot

Table 1. Speed-volume-density descriptive statistics at peak and off-peak periods

DESCRIPTIVES						ANOVA					
Section	N	Mean	SD	SE	Max	Comparison	*df*	MS	F	*p-value*	*F crit*
Speed (Off Peak)	4301	26.8	4.18	0.06	35.6	Between Groups	1	110143.26	5961.16	0	3.84
Speed (Peak)	1771	17.4	4.57	0.11	34.8	Within Groups	6070	18.48			
Total	6072	17.5	4.59	0.26	35.6	Total	6071				
Volume (Off Peak)	4301	431	349	5.32	1209	Between Groups	1	578161528	6063.47	0*	3.84
Volume (Peak)	1771	1110	177	4.22	1762	Within Groups	6070	95351.66			
Total	6072	628	396	4.23	1762	Total	6071				
Density (Off Peak)	4301	26.8	24.6	0.38	193.7	Between Groups	1	2294876	3554.32	0	3.84
Density (Peak)	1771	69.6	24.6	0.65	262	Within Groups	6070	645.66			
Total	6072	35.4	29.2	0.31	262	Total	6071				

*The mean difference is significant at the 0.05 level.

Table 2. V-K-Q descriptive statistics by rainfall intensity for peak periods

DESCRIPTIVES							ANOVA				
Section	Intensity	N	Mean	SD	SE	Max	Comparison	*df*	MS	F	*p-value*	*F crit*
Speed (mph)	None	1276	17.25	4.61	0.13	34.8	Between Groups	3	71.362	3.36	0.02	2.6
	Light	399	17.98	4.45	0.21	34.7	Within Groups	1514	21.254			
	Moderate	89	16.8	4.24	0.8	25.2						
	Heavy	7	15.57	2.55	0.96	17.8						
Total		1771	17.06	4.6	0.26	34.8	Total	1770				
Volume (veh/hr)	None	1276	1117	179.5	5.02	1762	Between Groups	3	84426.30	2.69	0.05	2.6
	Light	399	1089	173.9	8.70	1558	Within Groups	1767	31388.54			
	Moderate	89	1102	156.7	16.55	1453						
	Heavy	7	1059	193.9	73.29	1283						
Total		1771	1005	264.3	6.78	1762	Total	1770				
Density (veh/mi)	None	1276	70.62	27.5	0.77	262	Between Groups	3	2027.3	2.74	0.04	2.6
	Light	399	67.22	28.04	1.4	220	Within Groups	1767	741.12			
	Moderate	89	65.54	19.03	2.02	146						
	Heavy	7	57.41	9.62	364	72.9						
Total		1771	63.08	28.76	0.74	262	Total	1770				

*The mean difference is significant at the 0.05 level.

Table 3. V-K-Q descriptive statistics by rainfall intensity for off-peak periods

DESCRIPTIVES							ANOVA				
Section	**Intensity**	**N**	**Mean**	**SD**	**SE**	**Max**	**Comparison**	***df***	**MS**	**F**	***p-value***	***F crit.***
Speed (mph)	**None**	**3176**	**27.12**	**4.03**	**0.07**	**35.6**	**Between Groups**	**3**	**532.86**	**31.11**	**7E-20***	**2.6**
	Light	**901**	**25.79**	**4.53**	**0.15**	**35.1**	**Within Groups**	**4295**	**17.13**			
	Moderate	**217**	**25.64**	**4.33**	**0.29**	**31.3**						
	Heavy	**5**	**30.22**	**2.41**	**1.08**	**30.7**						
Total		**4299**	**27.14**	**3.85**	**0.05**	**35.6**	**Total**	**4298**				
Volume (veh/hr)	**None**	**3176**	**434**	**350.4**	**6.22**	**1204**	**Between Groups**	**3**	**81358.3**	**0.67**	**0.571**	**2.6**
	Light	**901**	**427**	**351.6**	**11.71**	**1209**	**Within Groups**	**4295**	**121659**			
	Moderate	**217**	**406**	**310.9**	**21.11**	**1057**						
	Heavy	**5**	**313**	**334.5**	**149.5**	**707**						
Total		**4299**	**403**	**409.7**	**6.07**	**1209**	**Total**	**4298**				
Density (veh/mi)	**None**	**3176**	**27.48**	**25.31**	**0.45**	**194**	**Between Groups**	**3**	**1987.61**	**3.288**	**0.02***	**2.6**
	Light	**901**	**25**	**22.64**	**0.75**	**117**	**Within Groups**	**4295**	**604.425**			
	Moderate	**217**	**24.21**	**21.51**	**1.46**	**146**						
	Heavy	**5**	**21.48**	**18.39**	**8.22**	**42**						
Total		**4299**	**24.5**	**28.93**	**0.43**	**194**	**Total**	**4298**				

*The mean difference is significant at the 0.05 level.

temperatures also resulted in 6.6% more traffic volume. Summarily, at peak hours, higher temperatures produced a corresponding increase in density and volumes, and a significant reduction in traffic speed. Table 3 shows that higher temperature consistently drove down the traffic volume and density, with hot temperatures exceeding 20 °C resulting in a 36.8% reduction in traffic volume. Conversely, higher temperatures resulted in greater average speeds to the tune of 15.7% at hot temperatures. As can be seen from Table 5, all traffic flow parameters are statistically significant at the *0.05* level, implying that there is a relationship between temperature and traffic flow parameters.

Table 4. V-K-Q descriptive statistics by temperature intensity for off-peak periods

		DESCRIPTIVES						ANOVA				
Section	Intensity	N	Mean	SD	SE	Max	Comparison	*df*	MS	F	*p-value*	*F crit*
Speed (mph)	Freezing	416	24.09	4.09	0.20	33.5	Between Groups	4	2664.81	177.6	6E-141*	2.4
	Cold	1366	25.32	4.33	0.12	35.6	Within Groups	4300	15.008			
	Normal	1411	27.46	3.96	0.11	31.3						
	Warm	905	28.72	3.00	0.10	30.2						
	Hot	207	28.59	2.92	0.20	28.9						
Total		4305	26.77	4.6	0.26	35.6	Total	4304				
Volume (veh/hr)	Freezing	416	1117	179.5	5.02	1762	Between Groups	4	84426.3	2.69	0.045*	2.4
	Cold	1366	1089	173.9	8.70	1558	Within Groups	4300	3138.5			
	Normal	1411	1102	156.2	16.55	1453						
	Warm	905	1059	193.9	73.29	1283						
	Hot	207	706	297.5	20.68	1115						
Total		4305	1005	264.3	6.78	1762	Total	4304				
Density (veh/mi)	Freezing	416	70.62	27.5	0.77	262	Between Groups	3	2027.3	2.74	0.042*	2.4
	Cold	1366	67.22	28.04	1.4	220	Within Groups	4300	741.12			
	Normal	1411	65.54	19.03	2.02	146						
	Warm	905	57.41	9.62	364	72.9						
	Hot	207	48	23.64	1.64	119						
Total		4305	63.08	28.76	0.74	262	Total	4304				

*The mean difference is significant at the 0.05 level.

4.3 Comparison and Discussion with Respect to Other Studies

Over the past decades, many studies have investigated the speed-flow-density relationships under rainy conditions and such studies also revealed that rainfall intensities have differing impacts on traffic flow parameters [2, 8, 13, 17–22]. The summary of the result of this comparison is presented in Table 6. A total of ten studies were compared. A quick look at Table 6 shows variation in the reduction of speeds across the studies, which is attributable to the differences in the weather, traffic speeds, driving behaviour, etc.

Most studies report that light rain has the smallest impact on traffic, which supports the notion that intensity matters when considering the impact of rainfall on traffic. In addition, studies reported that temperature had a near negligible impact on traffic speed [6, 13], although the reverse was the case in this study. This is arguably attributed to the fact that the studies compared in Table 6 were all freeways/motorways, and thereby would be affected differently by temperature levels. However, as Tables 4 and 5 show, there is a significant relationship between temperature level and traffic flow parameters and the impacts vary at peak and off-peak periods. As previously stated, this is the first

Table 5. V-K-Q descriptive statistics by temperature intensity for peak periods

DESCRIPTIVES							ANOVA					
Section	Intensity	N	Mean	SD	SE	Max	Compar-ison	*df*	MS	F	*p-value*	*F crit.*
Speed (mph)	Freezing	150	20.6	3.95	0.32	32.9	Between Groups	4	2041.18	125	5E-94*	2.4
	Cold	534	19.75	4.52	0.20	34.8	Within Groups	1766	16.33			
	Normal	540	16.75	4.23	0.18	31.8						
	Warm	388	14.80	3.53	0.18	27.9						
	Hot	159	15.09	2.67	0.21	22.0						
Total		1771	17.40	4.57	0.11	34.8	Total	1770				
Volume (veh/hr)	Freezing	150	1053	264.7	21.61	1541	Between Groups	4	231290	7.45	6E-06*	2.4
	Cold	534	1094	187	8.09	1596	Within Groups	1766	31025.9			
	Normal	540	1132	153.2	6.59	1625						
	Warm	388	1117	160.4	8.15	1762						
	Hot	159	1127	140.1	11.11	1444						
Total		1771	1110	177.4	4.22	1762	Total	1770				
Density (veh/mi)	Freezing	150	54.05	23.48	1.92	200	Between Groups	3	40291.5	61.63	8E-49*	2.4
	Cold	534	59.37	22.62	0.98	203	Within Groups	1766	653.72			
	Normal	540	73.16	26.75	1.15	237						
	Warm	388	81.00	28.69	1.46	230						
	Hot	159	78.16	24.56	1.95	262						
Total		1771	69.55	27.26	0.65	262	Total	1770				

*The mean difference is significant at the 0.05 level.

Table 6. Comparison of changes in operating speeds from several studies

Source (year)	Road Type	Location	Speed Limit (mph)	Intensity	Impact on Speed %					
					Rainfall			Temperature		
					Peak	O-Peak	N/S	Peak	O-Peak	N/S
Ibrahim and Hall (1994)	Freeway	Mississauga, Ontario, USA	62	Light			-12.9			N/A
				Heavy			-9.7			N/A
Kyte et al., (2001)	Freeway	Idaho, USA	70	Light			-13.6			N/A
				Heavy			-13.6			N/A
Smith et al., (2004)	Freeway	Virginia, USA	62	Light			-6.5			N/A
				Heavy			-6.5			N/A
Maze et al., (2005)	Freeway	Iowa, USA	70	Light			-2			-1
				Medium			-4			-1
				Heavy			-6			-2
Agarwal et al., (2006)	Freeway	Minnesota, USA	70	Light/Warm			-2.5			-1.5
				Heavy/Cold			-7			-3.6
Unrau and Andrey (2006)	Urban Expressway		55	Light			-10			N/A
				Medium			-10			N/A
				Heavy			-10			N/A
Rakha et al., (2008)	Freeway	Multiple, USA	87	Light			-3.6			N/A
				Heavy			-9			N/A
Billiot et al., (2009)	Motorway	France	80	Light			-8			N/A
				Heavy			-12.6			N/A
Tsapakis et al., (2013)	Urban	London, UK	70	Light			-2.1			-1.9
				Medium			-3.8			-3.2
				Heavy			-6			-3.2
This study (2018)	Urban Arterial	Manchester, UK	30	Light/Cold	+4.2	-4.9		-4.12	+5.1	
				Moderate/Normal	-2.6	-5.5		-18.7	+14	
				Heavy/Warm	-9.7	+11.4		-28.2	+19.2	
				Hot Temp	N/A	N/A		-26.7	+18.7	

study that considers urban city traffic (i.e. speeds at 30 mph), as opposed to other studies analyzing freeway, highway or urban freeway traffic. For instance, although [6] claim the study to be carried out in urban traffic, in reality it has been conducted on an urban motorway (given the speed limit of 70 mph). Furthermore, this study also categorized the traffic into peak and off-peak periods, enabling a proper understanding and appreciation of the traffic state.

5 Conclusion

This study has quantified the effect of weather conditions on traffic flow characteristics. It contributes to the understanding of how different rainfall intensities and temperature affect urban traffic. The key insight derived from this study is that rainfall actually affects urban traffic, but is dependent on the intensity, and peak/off-peak hours. Light rainfall had no impact on traffic (Table 2). However, average speed reduces with Moderate and Heavy rainfall by 2.6% and 9.7% respectively. The results are however different for off-peak periods, as can be seen from Table 3. Tables 4 and 5 present the off-peak/peak descriptive statistics respectively. This research provides policymakers with additional information towards developing effective traffic management solutions to improve congestion. Further research work will investigate the effect of weather-related data on weekends and holidays, and use of a larger dataset spanning 4 or more years that guarantee an even distribution of rainfall and temperature values enabling increases in results accuracy.

References

1. Cools, M., et al.: Changes in travel behavior in response to weather conditions: do type of weather and trip purpose matter? Transp. Res. Rec.: J. Transp. Res. Board **2157**, 22–28 (2010)
2. Maze, T., Agarwai, M., Burchett, G.: Whether weather matters to traffic demand, traffic safety, and traffic operations and flow. Transp. Res. Rec.: J. Transp. Res. Board **1948**, 170–176 (2006)
3. Goodwin, L.C.: Weather Impacts on Arterial Traffic Flow. Mitretek Systems Inc. (2002)
4. Rämä, P.: Effects of weather-controlled variable speed limits and warning signs on driver behavior. Transp. Res. Rec.: J. Transp. Res. Board **1689**, 53–59 (1999)
5. Hogema, J.: Effects of Rain on Daily Traffic Volume and on Driving Behaviour (1996)
6. Tsapakis, I., Cheng, T., Bolbol, A.: Impact of weather conditions on macroscopic urban travel times. J. Transp. Geogr. **28**, 204–211 (2013)
7. Jones, E.R., Goolsby, M.E.: The environmental influence of rain on freeway capacity. Highw. Res. Rec. (321), 74–82 (1970)
8. Smith, B.L., et al.: An investigation into the impact of rainfall on freeway traffic flow. In: 83rd Annual Meeting of the Transportation Research Board, Washington DC. Citeseer (2004)
9. Ibrahim, A.T., Hall, F.L.: Effect of adverse weather conditions on speed-flow-occupancy relationships (1994)
10. Mahmassani, H.S., et al.: Incorporating weather impacts in traffic estimation and prediction systems. US Department of Transport, Washington, vol. 108 (2009)
11. Wang, L., et al.: An analysis of effects of rainfall on travel speed at signalized surface road network based on probe vehicle data. In: ICTTS, Xian, China, pp. 2–4 (2006)
12. Theofilatos, A., Yannis, G.: A review of the effect of traffic and weather characteristics on road safety. Accid. Anal. Prev. **72**, 244–256 (2014)
13. Agarwal, M., Maze, T.H., Souleyrette, R.: Impacts of weather on urban freeway traffic flow characteristics and facility capacity. In: Proceedings of the 2005 Mid-Continent Transportation Research Symposium (2005)

14. Akin, D., Sisiopiku, V.P., Skabardonis, A.: Impacts of weather on traffic flow characteristics of urban freeways in Istanbul. Procedia-Soc. Behav. Sci. **16**, 89–99 (2011)
15. Koetse, M.J., Rietveld, P.: The impact of climate change and weather on transport: an overview of empirical findings. Transp. Res. Part D: Transp. Environ. **14**(3), 205–221 (2009)
16. Qiu, L., Nixon, W.: Effects of adverse weather on traffic crashes: systematic review and meta-analysis. Transp. Res. Rec.: J. Transp. Res. Board **2055**, 139–146 (2008)
17. Rakha, H., Crowther, B.: Comparison of Greenshields, Pipes, and Van Aerde car-following and traffic stream models. Transp. Res. Rec.: J. Transp. Res. Board **1802**, 248–262 (2002)
18. Billot, R., El Faouzi, N.-E., De Vuyst, F.: Multilevel assessment of the impact of rain on drivers' behavior: standardized methodology and empirical analysis. Transp. Res. Rec.: J. Transp. Res. Board **2107**, 134–142 (2009)
19. Kyte, M., et al.: Effect of weather on free-flow speed. Transp. Res. Rec.: J. Transp. Res. Board **1776**, 60–68 (2001)
20. Agarwal, M., Maze, T., Souleyrette, R.: The weather and its impact on urban freeway traffic operations. In: Proceedings of the 2005 Mid Continent Transportation Research Symposium (2006)
21. Unrau, D., Andrey, J.: Driver response to rainfall on urban expressways. Transp. Res. Record **1980**(1), 24–30 (2006)
22. Rakha, H., et al.: Inclement weather impacts on freeway traffic stream behavior. Transp. Res. Rec.: J. Transp. Res. Board **2071**, 8–18 (2008)

Fast Identification of Interesting Spatial Regions with Applications in Human Development Research

Carl Duffy[1], Deepak P.[2(✉)], Cheng Long[2], M. Satish Kumar[2], Amit Thorat[3], and Amaresh Dubey[3]

[1] Sungkyunkwan University, Seoul, Republic of Korea
cduffy38@qub.ac.uk
[2] Queen's University Belfast, Belfast, UK
deepaksp@acm.org, {cheng.long,s.kumar}@qub.ac.uk
[3] Jawaharlal Nehru University, New Delhi, India
{athorat,amareshdubey}@mail.jnu.ac.in

Abstract. Large-scale demographic datasets with spatial information provide a rich platform for human development research. Much emphasis is often placed on understanding deviations from dataset-level behavior across demographic attributes within spatially coherent regions, since those could point to a local condition worth addressing through regional policies, or at the other extreme, a less known success story that offers new learnings. Inspired by such scenarios, we build upon domain knowledge from HDR to devise an interestingness scoring for spatial regions and formulate the computational task of interesting spatial region identification. Accordingly, we develop a taxonomic organization of spatial regions and formulate bounds on interestingness scores, which are then leveraged to develop an efficient technique to address the task. Our search method is empirically evaluated over two real-world datasets, and is seen to record orders of magnitude of response time improvements over region enumeration. The absolute response times and the memory overheads of our approach are seen to be within highly desirable ranges, establishing the effectiveness of our solution for the task.

1 Introduction

Human development research (HDR) studies often target to draw insights from existence and non-existence of correlations between attributes of interest, as observed within a dataset. Recent efforts on collecting large general-purpose (secondary) datasets involving hours of interaction with each individual/household such as the Indian Human Development Survey (IHDS [2]) capture a wide variety of attributes and provide a platform for accelerating human development research through leveraging data science technologies. The IHDS dataset has been used for widely varying kinds of analysis of inter-attribute correlations; these include studying the relationship between height and cognitive achievement

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 408–416, 2018.
https://doi.org/10.1007/978-3-319-98812-2_37

in children [7] and studies involving relationship between energy access and poverty [4].

In a collaborative research project bringing together researchers from both data analytics and social sciences, we explore how data analytics can enable accelerate progress on HDR. Our intent is to develop efficient methods for exploratory data analytics tailored towards zeroing in on spatial regions and inter-attribute correlations that may be, prima facie, worthy of deeper study using conventional social science methods.

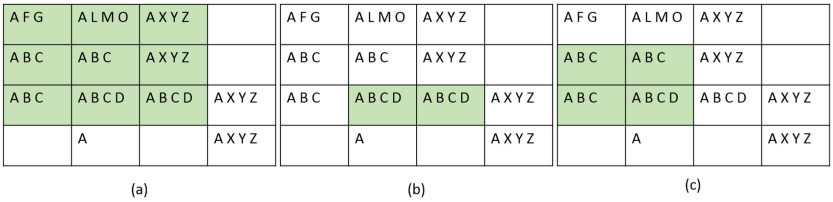


Fig. 1. Example for interesting region identification

Example: Consider a spatial dataset that is gridded into a 4×4 grid in Fig. 1. Within each grid cell, there could be one or more attribute pairs that exhibit a different trend than what is observed across the dataset. From a computational perspective, we are interested only in the set of such attribute pairs and not on actual attributes involved in the pair; accordingly, we use a boolean feature to denote each distinct attribute pair. The boolean feature is only shown in such regions where they are 'true' (i.e., deviant behavior is observed). Thus, the top-left grid cell in the dataset in Fig. 1 has recorded local behavioral deviations in three attribute pairs, denoted by A, F and G; on the other hand, the top right cell does not contain any local (statistically significant) behavioral deviations in any attribute pairs under consideration. Figures 1(a), (b) and (c) denote different spatial regions, indicated by shading, which could be candidates for *interesting spatial regions*; we limit our study to rectangular regions in this paper. The region in Fig. 1(a), while being a large region, has just one attribute pair, A, that is present in all regions; thus, while scoring well on the region size, it fares poorly on the number of attribute-pairs on which the component cells deviate together. Figure 1(b), on the other hand, fares poorly on the size of the region, while exhibiting coherent deviations across four attribute-pairs, A, B, C and D. Figure 1(c) is somewhat mid-way between the two earlier cases, with the region being sizeable enough, and there being three attribute-pairs, A, B and C, on which the component cells record similar deviations. In particular, it presents a trade-off between region size and deviation behavior, and is intuitively more desirable than the two extremes in terms of region interestingness.

Our Contributions: In this paper, we translate the aforementioned factors that characterize a family of HDR questions into a simple formulation of

interestingness for spatial regions. We then outline the computational problem of identifying the top interesting regions along with deviations that make the region interesting. We restrict our attention to rectangular regions regions in a gridded dataset, rectangular shapes being a simple and first step to ensure spatial coherence. We develop a method for efficient identification of top-k interesting regions, and illustrate that our method is able to achieve very fast response times for reasonably large datasets. The efficiency of our method as well as the effectiveness of our formulation advances the state-of-the-art in data analytics methods suitable for accelerating HDR.

2 Related Work

Single Type of Spatial Attributes: Research into identifying regions of interest based on a *single type* of boolean spatial attribute has been extensively explored both within the statistical community (e.g., SaTScan [5]) as well as the data mining community [8]. This has been the mainstream research direction in spatial interesting region detection. The broad framework is that there is a type of boolean attribute of interest, such as whether or not a geographically mapped person has a particular disease, and regions of interest are defined as those that are large enough while also exhibiting an unusually high or unusually low incidence of the disease. The different methods differ in terms of the kind of shapes, viz., circles, ellipses or arbitrary shapes, of regions that they are able to identify. To position this against our method, one could consider each boolean feature in our setting as a spatial attribute type; in our example in Fig. 1, the boolean feature type L is true only in one cell, whereas A is true in nine cells in our sixteen dimensional grid. Within that analogy, our task may obviously be seen to be targeting the identification of regions based on a different form of unusualness - viz., high frequency - of a plurality of binary feature types.

Variants: While single binary spatial attribute type has largely been the target of attention, there has been work on looking at variations of the problem. [9] extends the task definition to encompass two spatial attribute types. Another work, [6], shifts the focus from discovering regions to discovering attribute pairs, and seeks to identify attribute pairs that are unusually correlated spatially across the dataset. [3] extends the discovery process from attribute-pairs to attribute-sets and is particularly targeted at continuous attributes. It intends to discover spatial regions where sets of attributes together exhibit extreme values; an example output could be $[R, \{A \uparrow, B \downarrow, C \uparrow\}]$ indicating that a spatial region described by R has been found with an unusually high values of attribute A and C along with unusually low values of attribute B. Unlike [3] that focuses on approximating interestingness estimates computed over continuous attributes, we look at exact interestingness scoring over a dataset involving plurality of binary attributes. Some more variants of interesting spatial region discovery are covered in a survey [1].

3 Task Definition and Interestingness Scoring

Overview: Our overall task of interest may be seen as comprising two phases: (i) identifying, for each grid cell, the set of attribute pairs that deviate from the global behavior to arrive at a spatial dataset (e.g., Fig. 1 ignoring the shading) with boolean features (each denoting a distinct pair of HDR attributes), followed by (ii) identifying interesting spatial regions within that dataset along with a set of boolean features that characterize the region. We will consistently use the term *features* to refer to the boolean features, each of which stand for a pair of HDR-relevant demographic attributes. Since the focus of this paper is on the second phase, we now formally layout the task definition for that.

Task Definition: Consider a two-dimensional grid $\mathcal{G}$ of size $p \times q$; we will use G to refer to any grid cell within $\mathcal{G}$, and G_{ij} to refer to the grid cell at the i^{th} row and j^{th} column. In enumerating rows and columns, we start from the top and left; thus, the top-left grid cell is G_{00}. The set of binary features that appear in (i.e., are true in) cell G will be denoted as G_F. We now define a candidate spatial region as a 2-tuple, $C = [G_{ij}, G_{i'j'}]$, where the region C encompasses all cells in the rectangular region with G_{ij} as the top-left cell, and $G_{i'j'}$ as the bottom-right cell; $i' \geq i$ and $j' \geq j$ hold. We use $Cells(C)$ as a shorthand to denote all cells within the region. As an example, the region in Fig. 1(a) would be $[G_{00}, G_{22}]$.We now define C_F, the set of boolean features that characterize C, as an intersection of the boolean features across all cells within C, i.e., $C_F = \cap_{G \in Cells(C)} G_F$.

Our task of identifying interesting spatial regions is that of identifying the top-k spatial regions that score highest on an interestingness score $\mathcal{I}(C)$, among all possible spatial regions within the grid $\mathcal{G}$.

Interestingness of a Spatial Region: We now describe the interestingness quantification for our task. As outlined in the introduction, we would like the interestingness score, $\mathcal{I}(C)$ to be directly related to both (i) $|C_F|$, and (ii) $|Cells(C)|$. Additionally, to ensure that C_F captures most behavioral deviations within the grid cells in C, we would like to additionally ensure that most G_F for cells within C have a high overlap with C_F. We are now ready to outline our function:

$$\mathcal{I}(C) = \sum_{G \in Cells(C)} \frac{|C_F|}{|G_F|} \tag{1}$$

In short, our notion of interestingness computes, for every cell, G in C, the fraction of items in G_F that form part of each and every cell in the region (since C_F is computed as the intersection), and adds up the cell-specific terms across all cells in C. The existence of a term for each cell ensures the preference for larger regions, whereas the construction of C_F as the intersection ensures that large regions are considered only if there is enough overlap on their features. The interestingness of the rectangular regions from Fig. 1 is tabulated in Table 1.

Table 1. Interestingness of Rectangular Regions from Fig. 1

Region	C	C_F	Score computation	$\mathcal{I}(C)$
Figure 1(a)	$[G_{00}, G_{22}]$	$\{A\}$	$\frac{1}{3}+\frac{1}{4}+\frac{1}{4}+\frac{1}{3}+\frac{1}{3}+\frac{1}{4}+\frac{1}{3}+\frac{1}{4}+\frac{1}{4}$	**2.57**
Figure 1(b)	$[G_{21}, G_{22}]$	$\{A, B, C, D\}$	$1+1$	**2.00**
Figure 1(c)	$[G_{10}, G_{21}]$	$\{A, B, C\}$	$1+1+1+\frac{3}{4}$	**3.75**

4 Our Method

A brute force search enumerating all rectangular regions is quadratic on both the height and width of the grid, and thus is quartic on the grid edge size for square grids. In this section, we describe a search method for interesting region identification that estimates upper bounds on interestingness scores for partially seen regions, and employs early pruning of regions towards achieving much faster turnaround times for the task.

4.1 Phase One: From HDR Attributes to Binary Features

While there could be a variety of ways to identify attribute-pairs that exposit a different behavior in a grid cell as against their relationship in the entire dataset, we adopt a simple chi-square[1] test to arrive at the determination. In particular, we identify, for every grid cell, the set of attribute-pairs that exhibit a statistically significant relationship. From among those attribute-pairs, we exclude those that have a statistically significant relationship across the entire dataset, thus arriving at the set of attribute-pairs that exhibit deviant behavior in the grid-cell when contrasted against their dataset behavior.

4.2 Phase Two: Fast Identification of Interesting Spatial Regions

Hierarchical Organization of Spatial Regions. We first organize the space of all spatial regions in a structured fashion; within this structure, each region $[G_{ij}, G_{i'j'}]$ appears at a taxonomy of regions rooted at the single cell region corresponding to the top-left cell, i.e., $[G_{ij}, G_{ij}]$. We allow for three kinds of 'expansions' to form the taxonomy, defined as follows:

- **Downward Expansion (D):** A downward expansion extends a region downward, i.e., from $[G_{ij}, G_{i'j'}]$ to $[G_{ij}, G_{(i'+1)'j'}]$. The latter region contains additional row of $(j' - j + 1)$ cells as against the former.
- **Rightward Expansion (R):** $[G_{ij}, G_{i'j'}] \rightarrow [G_{ij}, G_{i'(j'+1)}]$
- **Right-Down Expansion (RD):** $[G_{ij}, G_{i'j'}] \rightarrow [G_{ij}, G_{(i'+1)(j'+1)}]$

[1] https://en.wikipedia.org/wiki/Chi-squared_test.

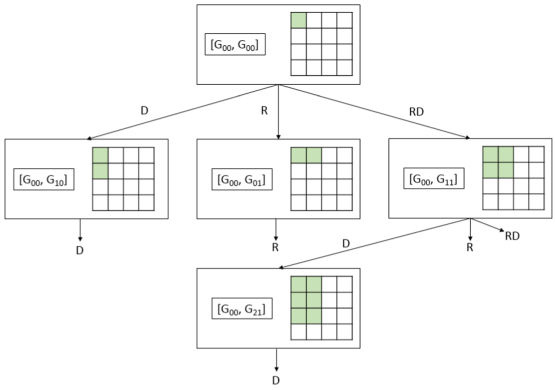


Fig. 2. Partial taxonomy rooted at $[G_{00}, G_{00}]$ illustrating expansion types.

A partial view of the taxonomy rooted at $[G_{00}, G_{00}]$ appears in Fig. 2, each node accompanied by an illustration of the spatial region it covers.

Restrictions on Expansions: If any sequence of R, D and RD expansions are allowed, a region may be encountered through multiple routes. We would like to avoid this for efficient search, and thus restrict the set of expansions. If a region is already taller than it is wide, we only allow it to be expanded into even taller regions through D expansions. Similarly, for regions that are wider than tall, they can only grow wider through R expansions. Square regions are allowed to grow on all three directions. Figure 2, it may be noted, depicts only the possible expansions under this set of restrictions.

Upper Bounds of Interestingness Scores. For every region, we would like to calculate the upper bound of interestingness scores that can be achieved by following each expansion type edge in the taxonomy. This needs to be computationally cheap to arrive at, while also being accurate enough to allow a search procedure relying on it to avoid exploring unpromising subtrees. One key property of our interestingness score is that a grid cell having no boolean feature being true in it is trivially uninteresting since it would push C_F to *null*; our bounds computation exploits this property.

D Subtree Upper Bound: We first identify the maximum number of D-type expansions that can be done on the region without having to include an empty cell in the region. Now, we consider the set of all such cells within the purview of that many D expansions; we will refer to this collection of cells as $MaxD(C)$. Each cell in $MaxD(C)$ can contribute a $|C_F|/|G_F|$ term to the interestingness score (Ref. Eq. 1). The maximum that a cell $G \in MaxD(C)$ can contribute is $\frac{min\{|C_F|,|G_F|\}}{|G_F|}$. This is so since the inclusion of G in the region could reduce the

features in the intersection to $min\{|C_F|, |G_F|\}$ or even lower due to the other cells that are brought in due to the expansion. Thus an upper bound for regions produced by all D expansions from C would be:

$$\mathcal{I}(C) + \sum_{G \in MaxD(C)} \frac{min\{|C_F|, |G_F|\}}{|G_F|} \tag{2}$$

However, this computation can be expensive when $MaxD(C)$ is large due to each cell having to be considered separately. Thus, we further coarsen the upper bound as:

$$UB_D(C) = \mathcal{I}(C) + |MaxD(C)| \times \frac{min\{|C_F|, min\{|G'_F||G' \in MaxD(C)\}\}}{min\{|G'_F||G' \in MaxD(C)\}} \tag{3}$$

It is easy to verify that $UB_D(C)$ is an upper bound of Eq. 2; we omit a formal proof for space constraints.

R Subtree Upper Bound: This bound is achieved by replacing $MaxD(C)$ above by $MaxR(C)$.

RD Subtree Upper Bound: Unlike R and D subtrees that are chains of R and D expansions respectively, the RD subtree would involve regions that can be reached by a sequence of RD expansions followed by R or D chains in addition to those that can be reached purely by RD expansions. Accordingly, $MaxRD(C)$ would be defined as:

$$\begin{aligned} MaxRD(C) = (ExpRD(C, maxrd(C)) - C) \cup \\ \bigcup_{1 \le i \le maxrd(c)} (MaxD(ExpRD(C, i) \cup MaxR(ExpRD(C, i)) \end{aligned} \tag{4}$$

where $maxRD(C)$ is the number of RD expansions possible without having to include an empty cell, and $ExpRD(C, n)$ indicates the spatial region obtained by n consecutive RD expansions from C. The RD upper bound then has a similar construction as for D, with $MaxRD(C)$ taking the place of $MaxD(C)$.

The Search Method. Having defined the construction of the taxonomy of spatial regions and upper bound computation for sub-trees in the taxonomy, we now outline a search method in Algorithm 2 for identifying top-k interesting spatial regions. The search process follows a best-first expansion approach exploring the space of spatial regions guided by the upper bounds. We use *top-k*$(\ldots)$ as a function that retains only the top-k regions in the supplied set based on their interestingness scores. The candidate set is progressively expanded through the search process, and the result set $\mathcal{R}$ is kept updated across the equations, and output at the end as the result set.

Algorithm 2. Search Method

Input: $p \times q$ Grid $\mathcal{G}$ populated with boolean features from Phase 1, k
Output: Top-k spatial regions based on interestingness

1: $\mathcal{R} = top\text{-}k(\{[G_{ij}, G_{ij}] | \forall\ i, j\})$ /*top-k interesting single-cell regions*/
2: $\mathcal{N} = \{[C, X, U] | C \in \{[G_{ij}, G_{ij}] | \forall\ i, j\}, X \in \{R, D, RD\}, U = UB_X(C)\}$
3: **while** $\exists N \in \mathcal{N}$ such that $N.U > min\{\mathcal{I}(C) | C \in \mathcal{R}\}$ **do**
4: $N = \arg\max_{N' \in \mathcal{N}} N'.U$
5:

$$E = \begin{cases} \{[Exp(N.C, D), D, UB_D(Exp(N.C, D))]\} & \text{if } N.X = D \\ \{[Exp(N.C, R), R, UB_R(Exp(N.C, R))]\} & \text{if } N.X = R \\ \{[Exp(N.C, RD), D, UB_D(Exp(N.C, RD))], & \\ \quad [Exp(N.C, RD), R, UB_R(Exp(N.C, RD))], & \text{if } N.X = RD \\ \quad [Exp(N.C, RD), RD, UB_{RD}(Exp(N.C, RD))]\} & \end{cases}$$

6: $\mathcal{N} = (\mathcal{N} - \{N\}) \cup E$
7: $\mathcal{R} = top\text{-}k(\mathcal{R} \cup \{Exp(N.C, N.X)\})$
8: $\mathcal{N} = \mathcal{N} - \{N | N \in \mathcal{N} \wedge N.U < min\{\mathcal{I}(C) | C \in \mathcal{R}\}\}$
9: Output $\mathcal{R}$

5 Experimental Evaluation

Datasets. Our task was motivated by the need for efficient bootstrapping of HDR over the IHDS dataset [2]. This dataset captures the data from over 200,000 individuals across hundreds of attributes in India, gridded into a 130×135 grid. Since the focus of our paper is on Phase 2, which could operate upon any spatial dataset, we collected a dataset of UK POI types from a popular community initiative[2], to form a second dataset for evaluation. This dataset contained 181k points of interest along with their types, which were placed on a 110×130 grid (14300 cells) using just their types.

Evaluation Measures. The improvement in response time against region enumeration is the primary measure of evaluation. Our method needs to maintain a set of candidates (i.e., $\mathcal{N}$) incurring a non-constant memory overhead, making that a factor of consideration. Thus, the maximum size of $\mathcal{N}$ across all iterations, denoted as $maxN$, is the memory requirement of our search method, which forms a second evaluation measure.

Evaluation on Response Times. Our method recorded significant improvements on response times on both datasets, with many orders of magnitude faster responses achieved on IHDS and upto 70 times faster responses on the POI dataset. In particular, the response times on the IHDS dataset are seen to be

[2] https://www.pocketgpsworld.com/modules.php?name=POIs.

of the order of hundreds of milliseconds, and sub-second response times are achieved on the POI dataset, indicating the effectiveness of our pruning and search method.

Memory Overhead. $MaxN$ was recorded as 53610 for IHDS and 56123 for POI. Using a generous estimate of 25 bytes to store each candidate, this corresponds to a memory overhead of just over 1 MB, a very modest requirement for modern machines.

6 Conclusions

We considered the problem of interesting spatial region identification over a dataset with a plurality of boolean attribute types, inspired by the task of bootstrapping human development research from a demographic dataset. We outlined a notion of interestingness and developed an efficient method for identifying top-k interesting rectangular spatial regions. Through empirical evaluation, we illustrated that our method achieves vast improvements over region enumeration with very modest memory overheads.

References

1. Deepak, P.: Anomaly detection for data with spatial attributes. In: Celebi, M.E., Aydin, K. (eds.) Unsupervised Learning Algorithms, pp. 1–32. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-24211-8_1
2. Desai, S., Vanneman, R.: National council of applied economic research, New Delhi. India human development survey (ihds) (2005). icpsr22626-v11. Ann Arbor, MI: Inter-university Consortium for Political and Social Research [distributor], pp. 02–16 (2016)
3. Eick, C.F., Parmar, R., Ding, W., Stepinski, T.F., Nicot, J.P.: Finding regional co-location patterns for sets of continuous variables in spatial datasets. In: SIGSPATIAL, p. 30. ACM (2008)
4. Khandker, S.R., Barnes, D.F., Samad, H.A.: Are the energy poor also income poor? evidence from India. Energy Policy **47**, 1–12 (2012)
5. Kulldorff, M.: A spatial scan statistic. Commun. Stat.-Theory Methods **26**(6), 1481–1496 (1997)
6. Shekhar, S., Huang, Y.: Discovering spatial co-location patterns: a summary of results. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 236–256. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-47724-1_13
7. Spears, D.: Height and cognitive achievement among Indian children. Econ. Hum. Biol. **10**(2), 210–219 (2012)
8. Telang, A., Deepak, P., Joshi, S., Deshpande, P., Rajendran, R.: Detecting localized homogeneous anomalies over spatio-temporal data. Data Min. Knowl. Discov. **28**(5–6), 1480–1502 (2014)
9. Wang, S., Huang, Y., Wang, X.S.: Regional co-locations of arbitrary shapes. In: Nascimento, M.A., et al. (eds.) SSTD 2013. LNCS, vol. 8098, pp. 19–37. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40235-7_2

Creating Time Series-Based Metadata for Semantic IoT Web Services

Kasper Apajalahti(✉)

Aalto University School of Science, Espoo, Finland
kasper.apajalaht@aalto.fi
https://seco.cs.aalto.fi/projects/mobile5g/

Abstract. In the near future, the Internet of things (IoT) will rapidly change and automate tasks in our everyday life. IoT networks have sensors measuring the environment and automated agents changing it with respect to predefined objectives. Modeling agents as web services requires lots of metadata from the environment in order to define the desired performance in a specific context. For this purpose, we propose an automatic measurement-based metadata creation method that analyses multivariate time series gathered from the sensors during agents change the environment. The time series analysis uses a cumulative sum algorithm (CuSum) to detect events and association rule learning to find temporal patterns. We evaluate our system with a Long-Term Evolution (LTE) simulator having mobile phones corresponding to IoT devices, LTE macro cells as the data source, and the Self-Organised Network (SON) functions as the automated agents in the network. Our experiments give promising results and show that the metadata creation process can be utilised to characterise IoT agents.

1 Introduction

Internet of Things (IoT) services combine sensors and IoT devices into applications that solve predefined use cases. Some services provide access to data gathered from sensors and others analyse the data and perform actions on the environment with respect to some objectives [8], for example, to increase the temperature of a room or signal quality of a mobile device. A key challenge in IoT is the interoperability between services and applications [5]. The services may be deployed to a variety of environments. Yet, the semantics of services should be defined with relevant metadata in order to find similarities in services across network domains. Analogously to a cellular network, one domain might want to prevent an anomalous behaviour by understanding how other domains have experienced and solved an issue in a similar context. For example, a sudden bad weather may be the root cause for terminals (IoT devices) to increase the load of nearby indoor cells (data sources) as people escape rain indoors. In this case, web services accessing local weather data combined with network-specific data and agent services would be necessary to act proactively to handle the upcoming load peak indoors.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 417–427, 2018.
https://doi.org/10.1007/978-3-319-98812-2_38

A general challenge with web services is the need to involve domain experts in developing the services [14]. Particularly, semantic IoT services have development costs in modeling the services with relevant metadata [12] and linking it to other ontologies in order to make services discovered and invoked [9]. The potential development costs also lead to a cold-start problem among the web service modelers: the SWS system cannot recommend suitable and interoperable services before developers in several domains have used their resources to manually model them. Hence, there is a need to automate the service development process.

In this paper, we propose a time series-based metadata creation process for semantic IoT services. The process is evaluated in a Long Term Evolution (LTE) network simulator environment having mobile devices that measure the quality of service and LTE macro cells that periodically report aggregated measurements. With respect to the LTE networks, the Self-Organised Networks (SON) has brought automation to the management tasks [11]. The SON functions can be viewed as specialised agents controlling the LTE cell configurations with respect to predefined objectives. The simulator contains simple SON functions which optimise the network performance and our goal is to characterise their behaviour with the metadata creation process.

The paper consists of the following parts. Related work is discussed in Sect. 2. After that, Sect. 3 gives an overview of the SWS methodology in the scope of IoT and cellular networks. Section 4 explains the theory and statistical methods used to create time series-based metadata for the services. Section 5 presents the experiments for the metadata creation process in an LTE simulator. Finally, Sect. 6 concludes the paper.

2 Related Work

Bytyçi et al. [4] propose a method that combines association rule learning and ontologies to mine patterns from water quality measurement data. They managed to enrich the mining results by first populating the context ontology with sensor data and then using the ontology as an input to the association rule learning.

Fan et al. [6] use association rule learning for sensor-based constructions to find contextual patterns from sensor measurements [6]. The experiments show that temporal patterns can be identified with respect to time metadata, such as a public holiday, weekday, or weekend.

Galushka et al. [7] examine data mining techniques for smart home environment. The authors present techniques both to transform time series into labeled segments and to use association rule learning to find temporal patterns from them.

Labbaci et al. [13] analyse web service logs and interactions between web services to learn frequent compositions and substitutions of services in a web service system. Their method has a similar focus on analysing the past data related to the web service invocations in order to learn characteristics of their

behaviour. Such as in our work, they use association rule learning to find the most frequent item sets from the web service-related data.

The related work focuses on association rule learning of measurement-based sensor data and web service logs. However, none of the earlier works learn association rules from sensor measurements in order to characterise IoT services, which is the focus of this paper.

3 Semantic Web Service Model

3.1 Methodology

The core idea in the Semantic Web Service (SWS) is to discover, compose, and invoke services with respect to user requests [2]. This idea applies both to IoT and cellular networks: requests based on the multivariate measurements of IoT devices or mobile phones need to be handled with an action that causes a desired impact to the measured values. Figure 1 describes a simplified web service ontology, where arrows depict *"hasElement"* relations. It also explains the analogy to IoT and cellular networks, such as the service corresponds to an agent, operation to an action, and effects to changes in the metric values. The architecture of the model is adapted from a simple SWS model, WSMO-lite [15]. A service has operations that aim to change the status of its environment.

Effects play a central role in using the ontology; the service model is used by linking the effects of requests and operations. For example, the goal of a network operation could be to enhance customer satisfaction for mobile users during a rush hour. This context-specific intention is mapped to some metric effects, such as an increase in the throughput and balancing the usage of physical resource blocks (PRBs). Based on the network measurements, some service operations are known to produce the requested context-specific effects and thus, they are mapped as responses to the request.

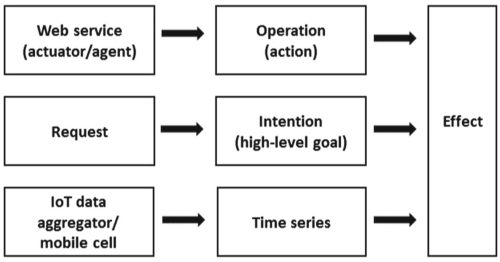


Fig. 1. Service ontology constructs for a service (top), request (middle), and environment/object status (bottom).

As domains may have dedicated data and means of management, there is a need to also understand the semantics of cross-domain effects. For example,

domains might have services that monitor threshold values for different key performance indicators (KPIs). Yet, some KPIs are associated (correlate) in the given context and are part of the same intention. Thus, both of the services would be valid solutions to a request having similar effect as an objective. More details about the dependency modeling of effects are defined in earlier research [3].

3.2 Cross-Domain Request

An example of a cross-domain user request from mobile network management is illustrated in Fig. 2. An operator from the network X wants to request better Quality of Service (QoS) in some context, such as during a public event. The requested intention can be achieved by increasing the throughput. Another network domain Y has deployed a SON function as a web service with an operation that is known to increase Reference Signal Received Power (RSRP) during a public event. Knowing the semantic similarity between the same cross-domain KPIs and that the throughput and RSRP in the domain Y are statistically associated, the given service operation can be discovered and mapped as a response to the user request. Altogether, even though a problem and solution would be in different networks and might address different parameters, the SWS system finds a request-operation mapping relation with respect to semantic modeling and statistical dependencies.

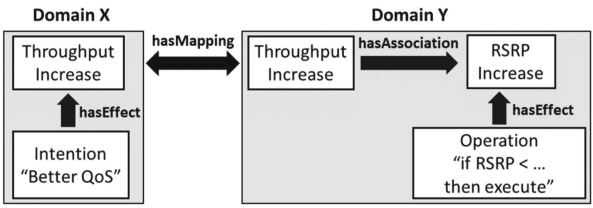


Fig. 2. An example of a cross-domain request-operation mapping.

In order to have a functional SWS system in a distributed and multi-domain sensor network, contextual metadata is needed, for example, the relevant operation-specific effects that enable the discovery of the web services. Creating and maintaining these manually is resource-intensive. Moreover, metadata mappings (such as corresponding KPIs) between different environments need to be resolved before the full benefit and utilization of the services can be realised. Automation for the service modeling and metadata addition can reduce the cost of deployment significantly by utilising SWS systems. In the next subsection, we introduce a process to address this problem with automatic metadata creation methods.

4 Methods to Create Service Metadata from Sensor Measurements

4.1 The Data Sequence of the Process

The measurement-based metadata is added as effects to the service operations and it is used to bind requests to operations. Our process analyses statistically the behaviour of a service operation in a given context while it is executed. Figure 3 illustrates the data sequence of processing metadata for a service operation based on realised actions and measured metric values both before and after the actions have been taken. In the beginning, the user decides how an operation is defined. For example, a service could be an algorithm and an operation a set of parameter values. The actions fulfilling this criteria are retrieved from the database (step 1) and based on their timestamps, a time series for every available metric of the operated object (such as the mobile cell) is retrieved from the measurement database (step 2). Time series are analysed with an event detection method (step 3) and the detected events are sent to the association learning component (step 4). This component detects whether one or more metrics have temporal correlation. Finally, the associated events are sent to the ontology and the service operation is populated with these events (step 5). The events are later used as service operation effects, as described in Fig. 1.

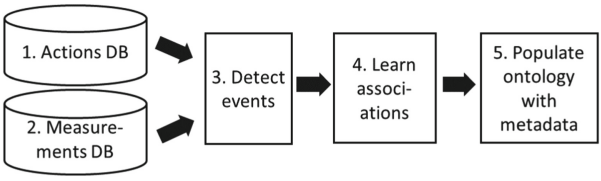


Fig. 3. Process flow for identifying associated events and adding them as operation metadata.

4.2 Event Detection with CuSum

The CuSum algorithm is a statistical quality control method that can be used to detect value changes in a time series. The basic concept is to cumulatively sum up changes between data points and a comparison value and flag a change if the sum exceeds a predefined threshold value. The Eq. 1 describes how to detect increasing event in our system. The equation contains a max of zero and the cumulative sum of value s_h, the data point x_t, and the combined comparison value of mean and standard deviation, μ and σ, calculated from the time series. σ is used as a threshold sum value for increasing trends.

$$s_h = max(0, s_h + x_t - \mu - \sigma) \tag{1}$$

For analysing decreasing trends in a time series, the Eq. 2 is used instead. Compared with the earlier equation, now a min operator is used and CuSum

contains a positive sign for the σ. The threshold sum for detecting a decreasing trend is $-\sigma$.

$$s_l = min(0, s_l + x_t - \mu + \sigma) \tag{2}$$

When CuSum is executed for all operation-specific actions, the outcome is a dataset where each row depicts a single action having a list of measurement events it produced. From this dataset, we may further learn operation-specific patterns between measurement events.

4.3 Temporal Pattern Mining with Apriori

Association rule learning is a data mining method that learns rules between the sets of items in a database. The idea is to analyse the co-occurrence of items in a database row and to use some measure and threshold to find out relevant rules. The simplest measure is the support, which is calculated as a proportion of the database rows containing the given set of items [10]. In our use case, the support is the proportion of detection timestamps containing a set of metric events. Thus, it indicates the frequency of the events occurring simultaneously in the given context.

In addition to support, confidence is another measure to determine associations between items. The Eq. 3 shows the definition of the confidence. It can be interpreted as an if/then pattern: if set of events X occurs, then set of events Y also occurs. As it can be seen, the measure indicates the proportion of X (the support of X) that also contains Y (the support of X and Y) [10].

$$conf(X \rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{3}$$

In this system, the objective is to learn support and confidence values for measurement events occurred during a set of actions made by an agent. For this purpose, we use an open-source implementation[1] that of a well-known Apriori algorithm (see [1] for further details).

5 Evaluation

5.1 Case Study: LTE Network Simulation with SON Functions

The applicability of our metadata creation process is evaluated with an LTE network simulator. The simulator environment comprises 20 LTE base stations with 32 LTE cells covering an area with a radius of 5 km. The simulator creates Performance Management (PM) data reports that contain cell level KPIs such as the average cell throughput, radio link failures (RLFs), average Reference Signal Received Power (RSRP), the overall usage of the Per Resource Blocks (PRB), and average channel quality indicator (CQI) level. The cell level KPIs are aggregations of the measurements made by the user equipments (UEs) that

[1] https://github.com/asaini/Apriori.

constantly report the experienced signal status to a cell they are attached to. The PM data of the cells are reported periodically in 15 min intervals in simulation time. The time series gathered from the PM data contains 10 time steps before and after the actions have been executed or activated.

Table 1 describes the scenarios created for our experiments. The idea was to create network issues with similar objectives but in different contexts; in all scenarios, users have issues getting the required throughput level, but the required actions differ significantly from each other.

Table 1. Simulation scenarios

Scenario	UEs	Objective	Solution
Coverage problem	1000	Inc. Thr.	Increase power
Local overload	350	Inc. Thr.	Downtilt
Mobile overload	500	Inc. Thr.	Balance load

In the coverage problem, the UEs are located uniformly in an area where the coverage is insufficient and the solution is to increase TXP to enhance the coverage and therefore the overall throughput level. The second scenario, local overload, has a few hundred UEs located in a small area near one base station hosting three cells. Now the throughput should be increased by adjusting the antenna tilt angles (remote electrical tilt, RET) towards the group of UEs. The third scenario, mobile overload, has 500 uniformly located background UEs and a group of 500 UEs constantly moving in the simulated area causing abrupt load peaks in the cells. An increase in the throughput in this case should be achieved by balancing the load between the nearby cells.

5.2 Context-Specific Support Values

Figure 4 presents the action-specific support values in different scenarios and KPIs. Figure presents one subplot for each scenario and each subplot presents KPI-specific support values for each action. Positive support value indicates a support measurement for an increase and negative a decrease. For example, the first five bars show support values for the increasing and decreasing events for the KPIs when no action has been taken in the coverage problem scenario. The first bar shows that increasing events for CQI has been measured with a support value of 0.12 and decreasing events with a value of 0.09. With respect to these experiments, a threshold level of ± 0.15 (marked with two dashed lines) is suitable for labelling action-specific KPI effects.

In general, we may conclude that the distribution of the scenario- and action-specific support values show that the detection of single KPI events works well as the values are plausible with respect to the actions. Especially, the throughput values show that the best agents in every scenario also enhance the throughput in the network, which is the desired outcome. Also, the fact that the number of

false positive support values (values when no action is taken) is low, indicates an adequate performance of the CuSum method.

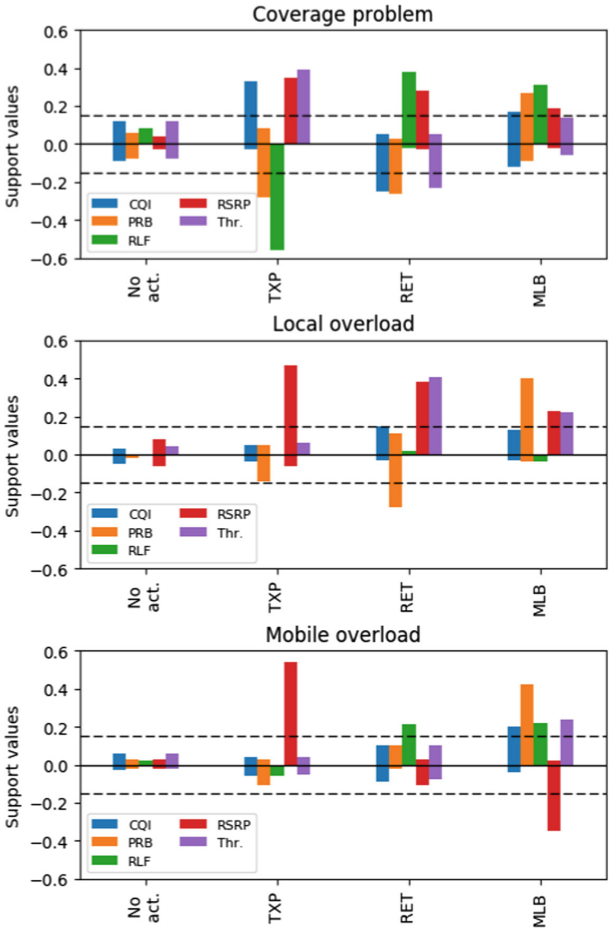


Fig. 4. Support values for action- and KPI-specific events in three scenarios.

5.3 Context-Specific Associations and Their Applicability as Metadata

The association rules for every scenario-specific action were generated with a minimum support level of 0.15 and confidence level of 0.70. Figure 5 shows the quantities of associations learned among the recorded events. With the given parameters, the best agents also generate the most associations between the KPI effects, whereas few associations are produced from other agents in the coverage problem. This is a desired outcome as our goal is to highlight the best matching agents in different contexts.

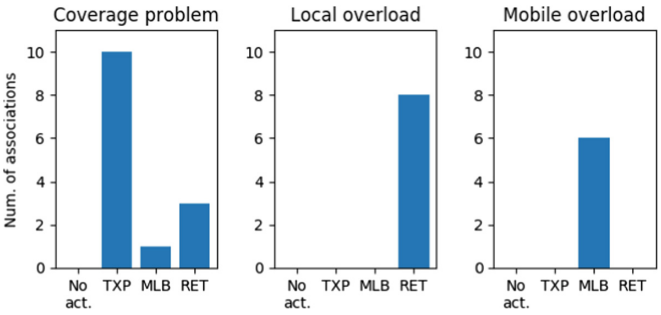


Fig. 5. Scenario-specific quantities of associations for service operations. Threshold for support is 0.15 and for confidence 0.70.

The final step in the metadata creation process is to populate the ontology instances with relevant events and associations. As defined in the Sect. 4.1, the populated ontology instances are web service operations: three scenario-specific operations for each web service (network agent). Finally, we may examine the request (ontology queries) that naps the queried effects with the relevant service operation metadata. Table 2 illustrates the examples of combining association rules that we tested and verified to retrieve correct mappings to the operations. For example, if a request contains associations from $IncTHR$ (throughput) to $DecRLF$ and from $DecRLF$ to $DecPRB$, it gives a unique mapping to the TXP agent operation on a coverage problem scenario. Similarly, the two other rows show rules that are also unique among all rule sets and that corresponds to the best solution in the scenario. In addition to the association rules shown in the table, the request query may include effects that pass the minimum support level (e.g. "increase the throughput") or negations of undesired effects (e.g."do not decrease the throughput").

Altogether, the demonstrated associations indicate that we managed to distinguish the important agent operations and scenarios from each other with our metadata creation process.

Table 2. Unique set of rules that characterise suitable agents in every scenario.

Scenario	Action	Matching rules
Coverage problem	TXP	$IncTHR \rightarrow DecRLF$, $DecRLF \rightarrow DecPRB$
Local overload	RET	$IncTHR \rightarrow DecPRB$, $IncRSRP \rightarrow DecPRB$
Mobile overload	MLB	$IncTHR \rightarrow IncPRB$, $DecRSRP \rightarrow IncPRB$

6 Conclusions and Future Work

We proposed a method of creating time series-based metadata for services that operate in IoT networks. The metadata creation process is a combination of statistical methods, event detection and association rule learning, and it is based on analysing multivariate time series gathered from the network elements while some actions (service operations) are executed. The process was evaluated with a Long Term Evolution (LTE) simulator where automated agents (web services) configure the antenna parameters of LTE macro cells in order to enhance the network quality. We created three simulation scenarios and evaluated the results of three agents in those. Our experiments show that the presented metadata creation process works on these scenarios; all suitable service operations can be characterised with the generated metadata. For future work, we examine and compare different event detection and pattern mining methods and evaluate them in more complex IoT environments.

Acknowledgment. We thank Vilho Räisänen and Eero Hyvönen for comments and support on early versions of this paper. This work is partially funded by Nokia.

References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of the 20th VLDB Conference, pp. 487–499 (1994)
2. Ankolekar, A., et al.: DAML-S: web service description for the semantic web. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 348–363. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-48005-6_27
3. Apajalahti, K., Niiranen, J., Kapoor, S., Räisänen, V.: Sharing performance measurement events across domains. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 463–469, May 2017. https://doi.org/10.23919/INM.2017.7987313
4. Bytyçi, E., Ahmedi, L., Kurti, A.: Association rule mining with context ontologies: an application to mobile sensing of water quality. In: Garoufallou, E., Subirats Coll, I., Stellato, A., Greenberg, J. (eds.) MTSR 2016. CCIS, vol. 672, pp. 67–78. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49157-8_6
5. Da Xu, L., He, W., Li, S.: Internet of things in industries: a survey. IEEE Trans. Ind. Inform. **10**(4), 2233–2243 (2014). https://doi.org/10.1109/TII.2014.2300753
6. Fan, C., Xiao, F., Yan, C.: A framework for knowledge discovery in massive building automation data and its application in building diagnostics. Autom. Constr. **50**, 81–90 (2015). https://doi.org/10.1016/j.autcon.2014.12.006
7. Galushka, M., Patterson, D., Rooney, N.: Temporal data mining for smart homes. In: Augusto, J.C., Nugent, C.D. (eds.) Designing Smart Homes. LNCS (LNAI), vol. 4008, pp. 85–108. Springer, Heidelberg (2006). https://doi.org/10.1007/11788485_6
8. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013). https://doi.org/10.1016/j.future.2013.01.010
9. Gyrard, A., Serrano, M., Atemezing, G.A.: Semantic web methodologies, best practices and ontology engineering applied to internet of things. In: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), pp. 412–417, December 2015. https://doi.org/10.1109/WF-IoT.2015.7389090

10. Hahsler, M., Grün, B., Hornik, K.: arules - a computational environment for mining association rules and frequent item sets. J. Stat. Softw. Articles **14**(15), 1–25 (2005). https://doi.org/10.18637/jss.v014.i15. https://www.jstatsoft.org/v014/i15
11. Hämäläinen, S., et al.: LTE Self-Organising Networks (SON): Network Management Automation. Wiley, Hoboken (2011)
12. Heß, A., Kushmerick, N.: Learning to attach semantic metadata to web services. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 258–273. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39718-2_17
13. Labbaci, H., Medjahed, B., Aklouf, Y.: Learning interactions from web service logs. In: Benslimane, D., Damiani, E., Grosky, W.I., Hameurlain, A., Sheth, A., Wagner, R.R. (eds.) DEXA 2017. LNCS, vol. 10439, pp. 275–289. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64471-4_22
14. Pautasso, C., Zimmermann, O., Leymann, F.: Restful web services vs. "big" web services: making the right architectural decision. In: Proceedings of the 17th International Conference on World Wide Web, WWW 2008, pp. 805–814. ACM, New York (2008). https://doi.org/10.1145/1367497.1367606
15. W3C: WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web, W3C Recommendation, World Wide Web Consortium (2010). https://www.w3.org/Submission/WSMO-Lite/. Accessed May 2018

Topic Detection with Danmaku: A Time-Sync Joint NMF Approach

Qingchun Bai[1], Qinmin Hu[2(✉)], Faming Fang[1], and Liang He[1]

[1] School of Computer Science and Software Engineering, East China Normal University, Shanghai, China
qchbai@ica.stc.sh.cn, {fmfang,lhe}@cs.ecnu.edu.cn
[2] Department of Computer Science, Ryerson University, Toronto, Canada
vivian@ryerson.ca

Abstract. Topic detection on web videos can effectively help collecting users' feedback and emotional tendency. With the features of relatively short, topic alignment and time synchronization, Danmaku comments can significantly extend the applications of topic detection. However, most of the current topic detection approaches fall short of considering the interior relation between adjacent time-steps which ignores the underlying temporal effects. To address this problem, we introduce a Joint Online Nonnegative Matrix Factorization model (JO-NMF) to detect latent topics with automatically exploiting Danmaku comments. Experimental results show great advantages of our proposed model on real-world Danmaku datasets. The results show our model outperforms baselines in topic detection with perplexity and RMSE for the noisy temporal data.

Keywords: Nonnegative matrix factorization · Danmaku
Web videos · Topic detection · Crowdsourcing

1 Introduction

Automatic topic detection and tracking (TDT) is a good solution to the effective organization of the online videos, which can generate topic words and predict the evolution for videos [1]. However, traditional topic detection models focus on the entire video-level comments which is limited to the intrinsic drawbacks: (1) it is difficult to capture the segment information since the comments are based on entire video; and (2) we do not know what specific content of the video causes the point of view, even if the generated topics can perfectly summarize the user's opinion for the video.

The emergence of Danmaku comments has significantly extended the applications of topic detection on videos [10]. Video based Danmaku is a scrolling marquee comments on videos, having originated in Niconico[1], and is exceedingly

[1] www.nicovideo.jp.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 428–435, 2018.
https://doi.org/10.1007/978-3-319-98812-2_39

popular among China video sites now [7]. Users can gain insightful information about the video with Danmaku, and this kind of comments can provide rich context information about the video.

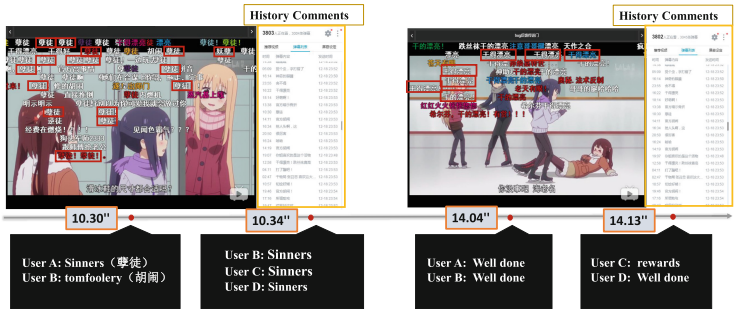


Fig. 1. A snapshot of Danmaku-enabled video. The video allows people to "shoot" their ideas while watching.

As shown in Fig. 1, user A posts a comment "Sinners" on the video at the time spot 10.30", while others follow the comments at different time spots, e.g., user B, C, and D at 10.34". This is a common phenomenon in Danmaku, which we call as "follower brush".

This kind of "follower brush" represents a typical character of time dependency, where the previous Danmaku comment has a great impact on the following Danmaku comments. Users keep commenting on video while watching, which forms a comment series over time. Then we believe that the topics among the comments series are also correlated over time. This motives us to model and capture time dependencies among video topics based on Danmaku comments.

Our work is related to NMF using temporal feature, since most of the existing NMF-based methods treat each underlying topic as a separate element while ignoring their correlations, thus limiting the evolution model's power of expression. In order to generate coherent topics at each time step, we learn the recent progress in NMF online and develop a new loss function based on a Joint Online NMF model (JO-NMF). We not only consider the content information of comments, but also the coherence of comments in a temporal dimension. JO-NMF incorporates the previous contextual information, and explicitly considers the time dependence to find out the underlying topics. When modeling documents internally, we can ensure the coherence of the topics in each time step.

The main contributions are listed as follows.

- A streaming JO-NMF algorithm is proposed based on normalized cumulative loss-gain that considers the temporal regularization and performs matrix factorization in each time slice.
- A real-world Chinese Danmaku corpus is crawled and built. The corpus leverages social context which can be used for topic detection and summarization

on online video. We recruit it to evaluate our proposed model, and the results show great advantages over the short and noisy temporal data.

2 Related Works

Research of TDT on Danmaku is comparatively rare, since Danmaku is an emerging comments type on videos in recent years. Most of existing research on Danmaku focused on exploring the potential value and motivation from the perspective of social interaction. Ma et al. [7] analyzed the usage of Danmaku through the users' views and motivations by semi-structured interviews. Yao et al. [10] discussed the Danmaku's potentialinality and proposed to use Danmaku in online video learning. He [3] explored the leading Danmaku and it's herding effect phenomena, which illustrated the unique characteristics of Danmaku.

Another piece of work was on exploring how to generate better tags for videos automatically, e.g., in [9], LDA based model was used to generate tags for the video. Lv et al. [6] proposed a temporal deep structured semantic model to label video. In [5], three features were designed for event detection on video.

Different to their works, we focus on time dependency to detect topic on Danmaku. To the best of our knowledge, we are the first attempt to consider temporal influence for topic detection on Danmaku.

3 Topic Detection Framework

3.1 Problem Formulation

Suppose there are V videos and their comment sets defined as $V = \{v_1, v_2, \ldots, v_{|V|}\}$. For each video v, we define a matrix $X^{(t)}$ with shape of $m \times n$ to represent a set of Danmaku comments at time interval t, where m represents the number of the comments and n denotes the dimension of the vocabulary. We observe a time step $t = \{1, 2, \cdots, T\}$ to formulate a comments stream $X = \{X^{(1)}, \cdots, X^{(T)}\}$, and the value of X is denoted by TF-IDF [4]. Our core insight is to find the top-K topics at each time step and generate topic lifelines which can represent the topics evolution.

3.2 Topic Detection via JO-NMF

As defined above, we aim to discover and extract coherent temporal topics from the Danmaku comments streams. We assume that the generation process is continuous, that is, the topic of the current t time step is developed from the previous t-1 time step, which is in conformity with the development in video-based comments. Based on such a hypothesis, our goal is to explore the evolutionary process of the continuous topics. A simple approach is to detect the topic independently, so the loss function is formulated as follows: $f\left(X^{(t)}\right) = \left\|X^{(t)} - W^{(t)}H^{(t)}\right\|_F^2$. Here $\|\cdot\|_F^2$ denotes the forbenius norm of Matrices.

However, such approach does not consider the intrinsic relationships between topics, decompose each $X^{(t)}$ directly will introduce noise due to the smoothness from $X^{(t-1)}$ to $X^{(t)}$ is not taken into account. The results may not be accurate especially when the datasets have no sufficient information. To address this issue, we would thus like to leverage prior knowledge to build a joint model for topic detection. Here, we considered joint factorization operations on multiple feature matrices generated at different times. For the current topic $H^{(t)}$ can satisfy both the distribution of historical data $X^{(1)}$ to $X^{(t-1)}$ and the present data $X^{(t)}$. In this paper, the joint factorization operations of multiple feature matrices are carried out by combining the accumulated data.

$$\begin{aligned} &\underset{W,H\geq 0}{\operatorname{argmin}} \sum_{j=1}^{t} (\left\| X^{(j)} - W^{(t)} H^{(j)} \right\|_F^2 + \alpha \left\| W^{(j)} \right\|_F^2 + \delta\left(t,j\right) \left\| H^{(j)} - H^{(j-1)} \right\|_F^2), \\ &s.t. \quad W^{(t)}, H^{(t)} \geq 0. \end{aligned} \tag{1}$$

Here the $W^{(j)}$ is the comment-topic matrix and $H^{(j)}$ is the topic-word matrix of the j-th time slice. We derive an algorithm with smoothness constraint to model the topic detection task and employ our time dependent JO-NMF method to enhance smoothness in W and H. In order to model how topics shifts along time, we apply the constraint on H, namely letting $h(H) = \left\| H^{(j)} - H^{(j-1)} \right\|_F^2$. The parameter δ shows how $H^{(j)}$ relates to $H^{(j-1)}$. The greater temporal distance between two documents, the less possibility that they describe the same topic, and the temporal weight should be set a relative smaller value.

The details are summarized as Algorithm 1.

Algorithm 1. JO-NMF for topic extraction

Require:
input $\{X^{(1)}, X^{(2)}, ..., X^{(T)}\}$, α, β,λ_t, μ, k, ϵ,
$k > 0, k \ll min(m,n)$,
$\alpha, \beta \leftarrow choose\ in\ [0.001, 0.5]$
Ensure:
The current time step decomposition results $W^{(t)} \in R_+^{m\times k}$ and $H^{(t)} \in R_+^{k\times n}$;
Initialize $W^{(0)}$, $H^{(0)}$ and k;
for each $t \in [1,T]$ **do**
  Initialize $W^{(t)}$, $H^{(t)}$ and $\delta^{'} = 1e-5$;// We use a parameter $\delta^{'} = 1e-5$ here;
  while $\left|\delta^{'} - \delta\right| \geq \epsilon$ **do**
    $\delta \leftarrow RMSEComputation(X^{(t)}; W^{(t)}; H^{(t)})$;
    if δ is ZERO **then**
      break;
    end if
    update $W_{ik}^{(t)}$ while fix $H_{kj}^{(t)}$ using updates rules:
    $W_{ik}^T \leftarrow \lambda_t W_{ik}^T \frac{\left[XW^T\right]_{ik}}{\left[WHH^T + \alpha W\right]_{ik}}$, $\forall i,k$;
    update $H_{kj}^{(t)}$ while fix $W_{ik}^{(t)}$ using updates rules:
    $H_{kj}^{(t)} \leftarrow \lambda_t H_{kj}^{(t-1)} \frac{\left[W^T X\right]_{kj}}{\left[WW^T H^{(t-1)} + \beta(t_i,t_j)\left(H - H^{(t-1)}\right)\right]_{kj}}$, $\forall k,j$;
  end while
end for

4 Experiment

4.1 Experimental Setup

Data Description. Data were collected from the Chinese video website Youku[2] and Bilibili[3]. We divide into four categories, namely Anime, Entertainment, Film and TV show. Danmaku contains crucial information such as "comments send time", "user ID", "timestamp corresponding to the video" and comment content. Meanwhile, 18% of user will "shoot" more than 3 comments and which means there exists long-tail users. The statistics formation for this data is shown in Table 1. We preprocess into a better data structure through the punctuation, segmentation, clean stop-words, noise reduction and other pre-processing steps. The Chinese words segmentation utilities used in this paper is JieBa word segmentation tool[4].

Table 1. The details of Danmaku comments datasets: (1) #video, #Danmaku, #users: the number of videos, Danmaku comments and uses respectively; (2) mean_v: the mean comments number per video; (3) mean_c: the mean length per comments and (4) max_user: The maximum number of an user.

Video category	#video	#Danmaku	mean_v	mean_c	#users	max_user
TV show	112	1.10E + 06	9.82E + 03	9.92	1.95E + 04	6905
Entertainment	101	3.38E + 05	3.35E + 03	10.72	1.02E + 04	56
Anime	68	1.26E + 04	1.85E + 02	10.12	4.02E + 03	646
Films	67	1.17E + 06	1.75E + 04	8.58	2.04E + 04	1760

Temporal Distribution of Danmaku. To better understanding the video-based Danmaku, the temporal distribution of Danmaku is analyzed. Here, we mainly discuss how to divide video-based Danmaku data into different time slices. In segmentation parts, we simply segment the comments by calculating the frequency. There is an example of comments frequency in Fig. 2. For the Anime datasets, the peaks correspond to the start. For the Entertainment datasets, the peaks correspond to multiple time steps for the video.

Baselines Methods. We compare our JO-NMF model with two other approaches: (1) **t-model** [4]. The t-model is standard NMF model. In t-model, the matrix factorization will be performing only in the first time slice, which is no longer change the distribution of the underlying topic vector. The realization of the NMF calculates method based on sparse constraint and multiplicative update rules of ℓ_1-norm regularization method. (2) **fix-model** [8]. The fix-model

[2] http://www.youku.com/.
[3] https://www.bilibili.com/.
[4] https://pypi.python.org/pypi/jieba/.

concerns the previous data. The difference with t-model is that the fix-model uses $W^{(t)}$ as the initialization matrix in the time slice t, which is, using the topic-word matrix $W^{(t-1)}$ of the previous time slice as the initial topic-word matrix of the t-th time slice.

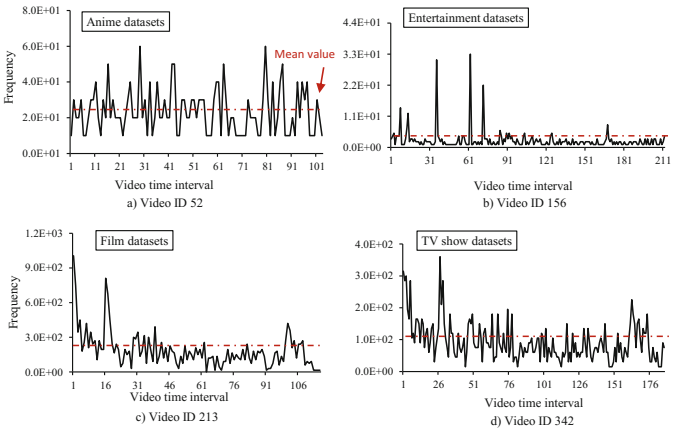


Fig. 2. Examples of temporal distribution. For Anime datasets, the peaks correspond to the start. For the Entertainment datasets, the peaks correspond to multiple time steps for the video. The red dotted line represents the mean frequency of the comments.

Evaluation Metrics. First, we report our results by perplexity metrics which measures the similarity between the term frequencies of the target document, where $perplexity = exp^{-(log(p(w_j)))/(N)}$ [2]. Here N represents the total Danmaku comments and $p(w_j)$ represents the probability of j-th word. The lower perplexity value means the better quality of the topic. Second, we measure the RMSE considers the case of forecasting [11]. For a set of data $\{x_1 \dots x_t\}$, we can use it to predict x_{t+1}.

4.2 Experimental Results

In this experiment, we first examine our models on four real world Danmaku datasets. We compare the various methods by setting different k, and the number varies among $(5, 10, 20, 30)$ for each dataset. The parameters used in the following experiments are as follows: we set $\lambda = 0.001$ and $\epsilon = 0.05$.

Perplexity Results. As shown in Table 2, JO-NMF is clearly outstrip t-model and fix-model. More specifically, the perplexity metrics of JO-NMF is 10 times better than others. This show that, the model is better to learn from the time dependent data. Moreover, the performance of t-model is the worst of the 3 methods. JO-NMF shows the advantages of modeling topics with temporal dependencies, which shows that the detection and filtering of topic can be performed well.

Table 2. The evaluation results of perplexity measure on real-world Danmaku comments datasets. The results are averaged values over 5 runs. The best results are shown in bold.

Perplexity								
	K = 5	K = 10	K = 20	K = 30	K = 5	K = 10	K = 20	K = 30
TV show datasets					Entertainment datasets			
t-model	2.78E + 08	4.77E + 09	5.35E + 12	2.00E + 17	4.77E + 04	3.43E + 07	2.56E + 14	1.45E + 15
fix-model	1.81E + 08	1.57E + 10	3.67E + 14	3.64E + 21	**1.28E + 04**	5.81E + 07	1.69E + 14	6.39E + 15
JO-NMF	**3.37E + 05**	**3.16E + 07**	**5.52E + 11**	**7.46E + 16**	1.74E + 05	**2.11E + 06**	**1.64E + 08**	**1.29E + 11**
Anime datasets					Film datasets			
t-model	2.539E + 13	3.503E + 16	2.73E + 15	8.69E + 21	5.73E + 07	1.43E + 09	3.72E + 13	4.56E + 17
fix-model	**1.923E + 13**	1.79E + 17	5.35E + 16	4.90E + 26	**1.12E + 07**	1.30E + 09	7.20E + 13	6.63E + 17
JO-NMF	1.78E + 14	**1.19E + 15**	**1.50E + 15**	**1.21E + 20**	8.49E + 07	**7.21E + 08**	**2.22E + 13**	**2.28E + 17**

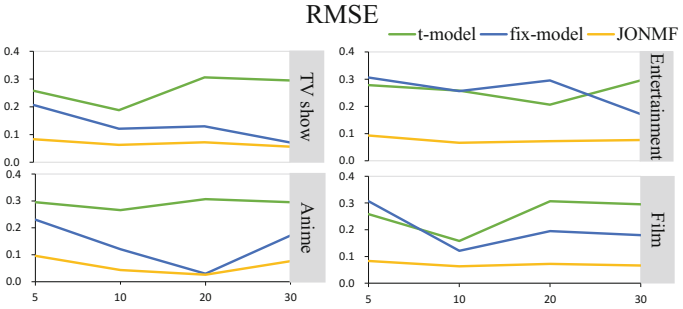


Fig. 3. RMSEs on the Danmaku datasets varying the topic k.

It can be proved that the distribution of topic-word and comment-topic are constantly changing at different time steps, using a fixed model detects the potential topic of the comments cannot achieve better results.

RMSE Results. We measured the RMSE considers the case of forecasting [11]. We collect $\sum X^{(t)}$ data as training periods and consider $t+1$ timestamp data $X^{(t+1)}$ as test periods. Figure 3 summarizes the comparison among other models for the task of forecasting varying the topic k. JO-NMF is relatively constant and clearly outperformed t-model and fix-model. Note that JO-NMF and fix-model was better than t-model, which might be because JO-NMF and fix-model consider the time-slice dependently. JO-NMF consistently better than others with different parameters k. Results shown that RMSEs are more correlative to the datasets, since the mean RMSE in Anime is 0.00898 which is 10 times higher than others (JO-NMF method achieves best performance).

5 Conclusions and Future Works

The Danmaku comments with rich context information have shown significant benefits for web video content analysis, and we specifically explore the effectiveness on topic detection using Danmaku comments. In this paper, we propose

a JO-NMF model for video topic detection, where we incorporate the previous data and explicitly considers the time decay to find out the latent topics. Experiments results show that the performance on topic detection is effectively improved by the proposed model.

As an emerging crowd-sourced comments, Danmaku have shown the power for video topic detection. The Danmaku comments are generated by the users, but different user's behavior introduces bias. In the future, we will consider to leverage the user's information to detect the community structure and the topic evolution road to different communities.

Acknowledgments. This research is funded by the Science and Technology Commission of Shanghai Municipality (No. 16511102702).

References

1. Cao, J., Zhang, Y., Ji, R., Xei, F., Su, Y.: Web video topics discovery and structuralization with social network. Neurocomputing **172**(C), 53–63 (2016)
2. Hayashi, K., Maehara, T., Toyoda, M., Kawarabayashi, K.I.: Real-time top-R topic detection on twitter with topic hijack filtering. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 417–426. ACM (2015)
3. He, M., Ge, Y., Chen, E., Liu, Q., Wang, X.: Exploring the emerging type of comment for online videos: Danmu. ACM Trans. Web **12**(1), 1 (2017)
4. Kalyanam, J., Mantrach, A., Saez-Trumper, D., Vahabi, H., Lanckriet, G.: Leveraging social context for modeling topic evolution. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 517–526 (2015)
5. Li, J., Liao, Z., Zhang, C., Wang, J.: Event detection on online videos using crowdsourced time-sync comment. In: International Conference on Cloud Computing and Big Data (2017)
6. Lv, G., Xu, T., Chen, E., Liu, Q., Zheng, Y.: Reading the videos: temporal labeling for crowdsourced time-sync videos based on semantic embedding. In: AAAI, pp. 3000–3006 (2016)
7. Ma, X., Cao, N.: Video-based evanescent, anonymous, asynchronous social interaction: motivation and adaption to medium. In: ACM Conference on Computer Supported Cooperative Work and Social Computing, pp. 770–782 (2017)
8. Vaca, C.K., Mantrach, A., Jaimes, A., Saerens, M.: A time-based collective factorization for topic discovery and monitoring in news. In: International Conference on World Wide Web, pp. 527–538 (2014)
9. Wu, B., Zhong, E., Tan, B., Horner, A., Yang, Q.: Crowdsourced time-sync video tagging using temporal and personalized topic modeling. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 721–730 (2014)
10. Yao, Y., Bort, J., Huang, Y.: Understanding Danmaku's potential in online video learning. In: CHI Conference Extended Abstracts on Human Factors in Computing Systems, pp. 3034–3040 (2017)
11. Yu, H.F., Rao, N., Dhillon, I.S.: Temporal regularized matrix factorization for high-dimensional time series prediction. In: Advances in Neural Information Processing Systems, pp. 847–855 (2016)

Data Mining

Combine Value Clustering and Weighted Value Coupling Learning for Outlier Detection in Categorical Data

Hongzuo Xu, Yongjun Wang(✉), Zhiyue Wu, Xingkong Ma, and Zhiquan Qin

National University of Defense Technology, 137 Yanwachi Street, Changsha, China
{xuhongzuo13,wangyongjun,maxingkong,tanzhiquan14}@nudt.edu.cn,
zhiyue.wu@outlook.com

Abstract. This paper introduces a novel unsupervised outlier detection method, namely WOD, for identifying outliers in categorical data. Existing subspace-based methods are challenged by overwhelming irrelevant features and their performance sometimes heavily depends on the setting of subspace size. Feature selection-based methods may omit the relevant information when removing features. In contrast, WOD works on value-level subspace exploring, i.e., separating irrelevant values based on value cluster structure, which avoids the dilemma of setting subspace size. Value outlierness is estimated by modeling weighted value couplings between relevant value set and value full set to further eliminate the interference from noisy features. We show that (i) WOD significantly outperforms five state-of-the-art outlier detectors on 12 real-world data sets with different levels of noisy features; (ii) WOD obtains good scalability.

1 Introduction

Outliers are rare and exceptional objects compared to the majority of normal objects. Outlier detection has extensive applications in various domains, e.g., network intrusion detection, insurance claim frauds, and medical diagnosis.

It is still a challenging problem to discover such outliers, especially in data with a mixture of relevant features and noisy features. Unsupervised outlier detection faces following major challenges (i) *Noisy features* (i.e., features in which outliers contain normal behaviours while some normal objects behave abnormally) dramatically downgrade the performance of outlier detectors by masking outliers as normal objects; (ii) Intricate *couplings* [7] (i.e., different types and hierarchies of interactions) between features or feature values greatly complicate the task of separating noisy features.

Most of existing unsupervised outlier detection methods (e.g., [2,9,15]) for categorical data are subspace-based methods. These methods are normally ineffective in handling data with noisy features and intricate couplings due to the lack of guidance related to outlierness scoring when searching the subspace. Feature selection is a widely-used approach to solve the problem of irrelevant

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 439–449, 2018.
https://doi.org/10.1007/978-3-319-98812-2_40

features. However, feature selection-enabled outlier detection methods (e.g., [12–14]) may easily neglect some relevant information when removing the features that contain not only noisy values but informative outlying values.

Based on the observation that outlying values normally have homophily couplings with each other [12,14,16], we attempt to estimate value outlierness by investigating the interactions between each value and an outlying value subset. This paper proposes a parameter-free outlier detection method, which *combines value clustering and Weighted value couplings based on Outlierness Diffusion*, WOD for short. WOD first constructs a value network based on pairwise value similarities and performs the community detection method on this value network to get the cluster structure. In order to get a relevant value set, WOD separates irrelevant values by identifying normal-value cluster and noisy-value cluster(s). Outlierness diffusion is modeled to investigate underlying value outlierness as value weight and cluster aggregation extent as cluster weight. WOD subsequently models weighted value couplings between each value and relevant values to obtain final value outlierness.

Extensive experiments show that (i) WOD significantly outperforms five state-of-the-art outlier detectors on 12 real-world data sets; (ii) WOD obtains good scalability w.r.t. data size and dimensionality.

2 Related Work

Traditional outlier detection methods like LOF, kNN and their multiple variants [6] are explicitly or implicitly dependent on some notion of distance. However, most proximity quantification (e.g. Euclidean distance function) are not meaningful or valid in categorical data [1]. In addition, these methods normally work on original full space, and thus they are ineffective when handling data sets containing noisy features.

Subspace-based outlier detection method is popularly proposed for data sets with irrelevant features in last decade. These methods [2,3,9,11,15,17] apply heuristic method or random search to identify relevant subspaces or outlying/normal patterns, and subsequently work on the subspaces to avoid the involvement of irrelevant features. However, these methods are easily misled by noisy features when searching the subspace due to the lack of guidance related to outlierness scoring, resulting in many faulty subspaces/patterns. The method proposed in [16] iteratively performs value selection and value outlierness scoring to jointly optimise both of these two phases. Nevertheless, its performance relies on its parameter setting (the subspace size), and the iteration may deviate from the right path if the initialisation cannot obtain a reliable estimation.

Feature selection has shown its effectiveness when handling data sets with noisy features. Although classification-oriented and clustering-oriented feature selection methods have been intensively studied [10], very limited work has been done on unsupervised feature selection for outlier detection [12–14]. Note that some features contain not only noisy values but also informative outlying values. These methods that work on the feature level may omit some relevant information when removing features.

3 Proposed Outlier Detector WOD

Let a data set be composed of a number of data objects $\mathcal{X}$ described by a set of features $\mathcal{F}$, where $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ is a set of data objects with size N and $\mathcal{F} = \{f_1, f_2, \cdots, f_D\}$ is a set of D categorical features. The value of feature f in object $\boldsymbol{x}$ is denoted by $v_f^{\boldsymbol{x}}$. Each feature has a possible feature value domain with countable size, i.e., $\mathcal{V}_f = \{v_1, v_2, \cdots\}$, where $f \in \mathcal{F}$. The full value set $\mathcal{V}$ is the union of all the feature domains, i.e., $\mathcal{V} = \cup_{f \in \mathcal{F}} \mathcal{V}_f$, where $\mathcal{V}_f \cap \mathcal{V}_{f'} = \emptyset, \forall f \neq f'$. We divide the full value set $\mathcal{V}$ into l clusters, i.e., $\mathcal{V} = \cup_{i=1}^{l} \mathcal{C}_i$, where $\mathcal{C} \cap \mathcal{C}' = \emptyset$, $\forall \mathcal{C} \neq \mathcal{C}'$.

The probability of value v from feature f is denoted by its frequency in all objects, i.e., $p(v) = \frac{|\{\boldsymbol{x} \in \mathcal{X} | v_f^{\boldsymbol{x}} = v\}|}{N}$ and the joint probability of two values v_i of feature f_i and v_j of feature f_j is $p(v_i, v_j) = \frac{|\{\boldsymbol{x} \in \mathcal{X} | v_{f_i}^{\boldsymbol{x}} = v_i \cap v_{f_j}^{\boldsymbol{x}} = v_j\}|}{N}$.

The proposed outlier detector WOD combines value clustering and weighted value coupling learning to estimate the outlierness of each value. As shown in Fig. 1, WOD first divides the full value set $\mathcal{V}$ into l clusters $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_l\}$. Similar values tend to gather together in same clusters. WOD then separates irrelevant values by identifying normal-value cluster $\mathcal{C}_{normal}$ and noisy-value cluster(s) $\bigcup \mathcal{C}_{noisy}$. Relevant values (outlying values) are retained in the remaining cluster(s) $\bigcup \mathcal{C}_{outlier}$. In order to resist the noise brought by irrelevant features, WOD only models the couplings between each value and outlying value set $\bigcup \mathcal{C}_{outlier}$. A value outlierness vector is derived from weighted summation of value couplings, which aims to highlight the couplings between outlying values. Object outlierness can be subsequently obtained by value outlierness.

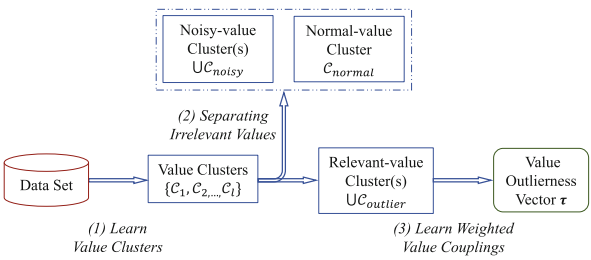


Fig. 1. The Proposed WOD Outlier Detector for Estimating Value Outlierness.

WOD's three phases (i.e., learning value clusters, separating irrelevant values, and learning weighted value couplings) are specifically introduced below.

3.1 Learning Value Clusters

WOD first constructs a undirected and weighted value network $G =< V, E >$ to perform community detection algorithm. Each node v in G is a feature value,

$v \in \mathcal{V}$. The entry in adjacency matrix $\mathbf{A}(v_i, v_j)$ is a edge weight between node v_i and node v_j:

$$\mathbf{A}(v_i, v_j) = \frac{p(v_i, v_j)}{\sqrt{p(v_i)p(v_j)}}, \tag{1}$$

where $\mathbf{A}(v_i, v_j) \in [0, 1]$ is Ochiai coefficient-based pairwise value similarity.

The quality of network depends on whether outlying values can be separated from normal values and noisy values. Ochiai coefficient is chosen because it is based on co-occurrence and individual frequency, which can well capture the characteristics of outlying values (i.e., infrequency and concurrency). Normal values are extremely frequent and rarely co-occur with outlying values. Although both noisy values and outlying values have low frequencies, they are supposed to co-occur randomly or follow a Gaussian distribution [14,16]. As a result, the edge weight between normal/noisy value and outlying value is small.

A modularity-based greedy optimisation method, namely Louvain [4], is employed to obtain the cluster structure of value network. It is chosen because of its effectiveness and efficiency. Most importantly, it can solve the problem of resolution limit thanks to its intrinsic multi-level nature [4].

3.2 Separating Irrelevant Values

Irrelevant values refers to normal values and noisy values. The cluster structure of value network is utilised to separate these two kinds of values.

Identification of normal-value cluster requires a rough estimation of value outlierness. Normal-value cluster has the lowest average outlierness.

Definition 1 (Normal-value Cluster). *Normal-value Cluster is identified by:*

$$\mathcal{C}_{normal} = \arg\min_{\mathcal{C}} \frac{1}{|\mathcal{C}|} \sum_{v \in \mathcal{C}} \delta(v), \tag{2}$$

where $\delta(v)$ is the outlierness of individual value v.

Inspired by [16], we use $\delta(v) = \frac{1}{2}(\frac{p(m)-p(v)}{p(m)} + \frac{p(b)-p(m)}{p(b)})$ to approximate value outlierness, where m is the mode value with the largest frequency of the corresponding feature containing value v, and b denotes the baseline value occurring most frequently among all the values in $\mathcal{V}$. $\delta(v)$ takes account the location parameter (i.e., mode) of the frequency distributions and use the same way to evaluate the outlierness of this location parameter as the base, which can make values from various frequency distributions semantically comparable.

A threshold of cluster size is set to identify noisy-value cluster(s).

Definition 2 (Noisy-value Cluster). *The size of noisy-value cluster $\mathcal{C}_{noisy}$ satisfies:*

$$|\mathcal{C}_{noisy}| < \alpha |\mathcal{C}_{max}|, \tag{3}$$

where α is an adjustment coefficient to regulate the threshold, and $\mathcal{C}_{max}$ is the cluster with the largest size among all the clusters except normal-value cluster.

The size of largest cluster $\mathcal{C}_{max}$ is used as a baseline. In terms of the setting of adjustment coefficient α, the size of normal-value cluster $\mathcal{C}_{normal}$ can be used since it determines the number of noisy values. If the proportion of normal-value cluster $\mathcal{C}_{normal}$ is large, the upper bound of noisy-value cluster is supposed to be small. The adjustment coefficient is set as $\alpha = 1 - \frac{|\mathcal{C}_{normal}|}{|\mathcal{V}|}$ to adjust the threshold of noisy-value cluster $\mathcal{C}_{noisy}$.

Except normal-value cluster $\mathcal{C}_{normal}$ and noisy-value cluster(s) $\bigcup \mathcal{C}_{noisy}$, the remaining cluster(s) contain(s) the relevant (outlying) values, which is denoted by $\bigcup \mathcal{C}_{outlier}$. $\mathcal{V}_o$ represents the set of values contained by $\bigcup \mathcal{C}_{outlier}$.

3.3 Learning Weighted Value Couplings

We define value weight and cluster weight by modeling outlierness diffusion, then model weighted value couplings to estimate the outlierness of each value.

Value weight and cluster weight are scored to highlight outlying values and clusters, which aims to capture the outlying-to-outlying value couplings. Note that Outlierness of a value is not only explicitly manifested by itself but can be demonstrated by relations with its neighbours (i.e., underlying outlierness). We score value weight based on the risk of infection from neighbour values. In terms of cluster weight, outlierness aggregation extent is employed because outlying values tend to connect tightly.

Definition 3 (Outlierness Diffusion-based Weight Scoring). *Value weight vector $\omega_{value} \in \mathbb{R}^{|\mathcal{V}|}$ and cluster weight vector $\omega_{cluster} \in \mathbb{R}^{l}$ are defined as follows:*

$$\omega_{value}(v) = \sum_{u \in \mathcal{V}_o} \psi(u, v)\delta(u), \tag{4}$$

$$\omega_{cluster}(\mathcal{C}) = \mathbb{I}_{\bigcup \mathcal{C}_{outlier}}(\mathcal{C}) \frac{1}{|\mathcal{C}|} \sum_{v \in \mathcal{C}} \phi(v, \mathcal{C})\omega_{value}(v), \tag{5}$$

where $\psi(u, v) = \frac{\mathbf{A}(u,v)}{\sum_{u' \in \mathcal{V}} \mathbf{A}(u,u')}$, $\phi(v, \mathcal{C}) = \frac{\sum_{v' \in \mathcal{C}} \mathbf{A}(v,v')}{\sum_{v' \in \mathcal{V}} \mathbf{A}(v,v')}$, and $\mathbb{I}$ is indicator function to set the weight of normal-value cluster and noisy-value cluster(s) as 0.

WOD builds a $|\mathcal{V}| \times |\mathcal{V}|$ coupling matrix to capture pairwise value couplings based on conditional possibility.

Definition 4 (Coupling Matrix). *Coupling matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ captures the interactions between values, which is defined as:*

$$\mathbf{M}(u, v) = p(v|u) = \frac{p(u, v)}{p(u)} \tag{6}$$

We employ conditional possibility because it is simple and is capable of fully describing the desired pairwise interactions. Outlying values are extremely infrequent and always concurrent with each other, and thus the conditional possibility between outlying values is larger. Therefore, conditional possibility can be utilised to distinguish the outlying-to-outlying couplings.

Based on coupling matrix, value outlierness is estimated by investigating the coupling strength with other values. WVC is short for Weighted Value Coupling.

Definition 5 (WVC-based Value Outlierness Scoring). *Value outlierness vector* $\tau \in \mathbb{R}^{|\mathcal{V}|}$ *is defined as:*

$$\tau(v) = \sum_{\mathcal{C}_i} \omega_{cluster}(\mathcal{C}_i) \sum_{u \in \mathcal{C}_i} \widetilde{\mathbf{M}}(v,u)\omega_{value}(u), \tag{7}$$

where $\widetilde{\mathbf{M}}$ *is a column-wise normalisation of* $\mathbf{M}$.

Note that value weight uncovers the underlying outlierness of each value, and cluster weight reflects the outlierness aggregation extent. These two kinds of weight capture two key characteristics of outlying values (i.e., infrequency and concurrency). A value is scored as high outlierness if and only if it has strong couplings with lots of heavy-weight values. Such weighted value coupling assists WOD to learn the complicated relations between features/values and facilitate the value outlierness estimation.

3.4 The Algorithm and Its Time Complexity

The procedure of WOD is presented in Algorithm 1. Steps (1–15) are performed to evaluate value outlierness by WOD. Following [12,16], Steps (18) employs

Algorithm 1. *WOD-based Outlier Detection*

Input: $\mathcal{X}$ - data objects
Output: R - an outlier ranking

1: Construct network adjacency matrix $\mathbf{A}$ and coupling matrix $\mathbf{M}$
2: Generate cluster structure $\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_l\}$ by Louvain algorithm
3: **for** $\mathcal{C}$ in $\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_l\}$ **do**
4: Identify $\mathcal{C}$ is $\mathcal{C}_{normal}$ or $\mathcal{C}_{noisy}$ using Equation (2) and (3)
5: **end for**
6: Initialise value weight vector $\omega_{value} \in \mathbb{R}^{|\mathcal{V}|}$ and cluster weight vector $\omega_{cluster} \in \mathbb{R}^{l}$
7: **for** $\mathcal{C}$ in $\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_l\}$ **do**
8: **for** v in $\mathcal{C}$ **do**
9: $\omega_{value}(v) \leftarrow \sum_{u \in \mathcal{V}_o} \psi(u,v)\delta(u)$
10: **end for**
11: $\omega_{cluster}(\mathcal{C}) \leftarrow \mathbb{I}_{\cup \mathcal{C}_{outlier}}(\mathcal{C}) \frac{1}{|\mathcal{C}|} \sum_{v \in \mathcal{C}} \phi(v,\mathcal{C})\omega_{value}(v)$
12: **end for**
13: **for** v in $\mathcal{V}$ **do**
14: $\tau(v) \leftarrow \sum_{\mathcal{C}_i} \omega_{cluster}(\mathcal{C}_i) \sum_{u \in \mathcal{C}_i} \widetilde{\mathbf{M}}(v,u)\omega_{value}(u)$
15: **end for**
16: Initialise $\mathbf{q} \in \mathbb{R}^{|\mathcal{X}|}$ as an outlierness vector for data objects
17: **for** $\boldsymbol{x}$ in $\mathcal{X}$ **do**
18: $\mathbf{q}(\boldsymbol{x}) \leftarrow \sum_{f \in \mathcal{F}} \tau(v_f^{\boldsymbol{x}})\omega_{feature}(f)$
19: **end for**
20: $R \leftarrow$ Sort $\mathcal{X}$ w.r.t. $\mathbf{q}$ in descending order
21: **return** R

weighted summation of value outlierness to calculate object outlierness, where $\omega_{feature}(f) = \sum_{v \in \mathcal{V}_f} \tau(v) / \sum_{v \in \mathcal{V}} \tau(v)$. An object outlierness ranking is subsequently generated in Step (20) by descending order according to the object outlierness score. Outliers are data objects with high ranking.

WOD requires one scanning over the whole data objects to acquire network adjacency matrix $\mathbf{A}$ and coupling matrix $\mathbf{M}$ in Step (1), which has $O(|\mathcal{X}||\mathcal{V}|^2)$. The time complexity of Louvain algorithm in Step (2) is $O(|\mathcal{V}| \log |\mathcal{V}|)$. Steps (3–5) separate irrelevant values, which has $O(l|\mathcal{V}|)$. Steps (6–15) takes $O(l|\mathcal{V}|)$ to estimate value outlierness. The object outlierness estimation and sorting in Steps (16–21) have $O(|\mathcal{X}||\mathcal{V}|)$. Therefore, WOD has linear time complexity w.r.t. data size and quadratic time complexity w.r.t. the number of features since the number of values per feature is countable (normally very small).

4 Experiments and Evaluation

4.1 Outlier Detectors and Parameter Settings

WOD is evaluated with five outlier detectors: coupling learning-based methods POP [16] and CBRW [12], subspace-based methods ZERO [15] and iForest [11], and a widely-used performance baseline for outlier detection LOF [5]. iForest and LOF can only process numerical data. The categorical data sets are converted into numerical data by one-hot encoding transformation method to allow iForest and LOF to perform on the same data sets.

WOD is a non-parameter method. Five competitors of WOD are performed with recommended settings. As recommended in [16], POP is processed with $k = 0.3$. Following [12], CBRW uses $\alpha = 0.95$. ZERO and iForest are used with the suggested settings in [11,15]. We use $MinPts = 5$ as the default setting of LOF. WOD[1] and its competitors are implemented in JAVA.

4.2 Data Sets

Twelve publicly available real-world data sets are used, which cover diverse domains, e.g., text classification, image object recognition, and health care, as shown in Table 1. These data sets are transformed from extremely imbalanced data, where the rare classes are treated as outliers versus the rest of classes are normal [8,12,14]. Only categorical features are used and features containing one value are removed because no information is carried by them to detect outliers.

4.3 Performance Evaluation Methods

A widely-used evaluation method, the area under ROC curve (AUC), is used to evaluate the effectiveness of five outlier detectors. The range of AUC is $[0, 1]$ and higher AUC demonstrates better accuracy. The AUC value would be around 0.5

[1] The code of WOD is available at https://sites.google.com/site/hongzuoxu/.

given a random ranking of data objects. *Wilcoxon* signed rank test is employed to check the significance of WOD performance versus its competitors. The runtime of all the methods is recorded for comparing their efficiency in scalability test.

Data Indicator is usually used to quantitatively capture the underlying characteristics of data sets, which has strong association with performance of outlier detectors. Two data indicators, proportion of noisy features η_{noisy} and maximum modularity Q_{max}, are defined to measure the complexity of each data set. Following [12,14], the AUC performance of using frequency histogram of each feature to detect outliers is employed to evaluate feature efficiency. η_{noisy} is the proportion of features that have corresponding AUC value smaller than 0.5. Q_{max} is the maximum modularity of the community structure obtained by LOUVAIN. A data set with high Q_{max} indicates that the partition of the network can clearly separate irrelevant values from outlying values. Their quantisation is reported in Table 1.

4.4 Effectiveness of WOD in Real-World Data Sets

The AUC performance of WOD and its five contenders is reported in Table 1. WOD achieves the best performance on eight data sets and performs close to the best on other four data sets (the AUC difference is less than 0.02). WOD significantly outperforms its five competitors at the 95% confidence level and averagely obtains more than 16%, 6%, 10%, 18%, and 62% AUC improvement than POP, CBRW, ZERO, iForest and LOF, respectively.

Table 1. Basic Information of Data Sets Used, Quantization Results of Data Indicators, AUC Performance of Six Outlier Detectors. Data is sorted by η_{noisy}. k indicates the percentage of $\mathcal{V}_o$ to the full value set $\mathcal{V}$.

Basic Data Info.					Data Indicators		Outlier Detectors					
Data	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{V}_o\rvert(k)$	$\lvert\mathcal{X}\rvert$	η_{noisy}	Q_{max}	WOD	POP	CBRW	ZERO	iForest	LOF
wap.wc	4229	8458	2487(29%)	346	99.01%	0.0053	0.9918	**1.0000**	0.7900	0.6552	0.5558	0.5161
SylvaA	172	344	86(25%)	13086	91.28%	0.0017	**0.9721**	0.7098	0.9310	0.8821	0.8136	0.4298
aPascal	64	128	51(40%)	12695	81.25%	0.0592	**0.8763**	0.7830	0.8190	0.6958	0.5117	0.5722
SylvaP	87	174	34(20%)	13086	78.16%	0.0042	**0.9818**	0.7635	0.9689	0.9585	0.9098	0.5429
Census	33	495	142(29%)	299285	57.58%	0.1945	**0.8220**	0.2804	0.6678	0.6420	0.5400	0.5924
HIVA	1617	3234	1576(49%)	4229	57.08%	0.0238	0.6828	0.6383	0.6915	**0.6930**	0.6858	0.4501
CelebA	39	78	38(49%)	202599	48.72%	0.0507	0.8936	**0.8968**	0.8462	0.7595	0.6742	0.4726
CT	44	88	25(28%)	581012	34.09%	0.0045	**0.9771**	0.9455	0.9703	0.9725	0.9348	0.5000
CAL16	253	506	225(44%)	829	15.02%	0.1011	**0.9932**	0.9928	0.9925	0.9878	0.9716	0.3881
Credit	9	77	30(38%)	30000	11.11%	0.3944	**0.6981**	0.4109	0.5804	0.6628	0.6612	0.5415
Arrhy	64	128	56(44%)	452	6.25%	0.0316	0.6859	0.6761	**0.6910**	0.6644	0.6883	0.6008
R10	100	200	100(50%)	12897	0.00%	0.0840	**0.9905**	0.9837	**0.9905**	0.9869	0.9789	0.9127
					Average		**0.8804**	0.7567	0.8283	0.7967	0.7438	0.5433
					p-value		-	0.0068	0.0137	0.0034	0.0024	0.0005

WOD separates irrelevant values based on cluster structure of value network and learns weighted value couplings to highlight outlying-to-outlying couplings,

which improves the resilience of WOD and enables it to handle data sets with different proportion of noisy features. Q_{max} indicates the quality of value network partition. WOD is more likely to perform well on the data sets with high Q_{max}, e.g., *Credit* and *Census*. k reported in Table 1 can be used to illustrate the relevant value proportion of different data sets. POP employs selective value couplings with fixed selective rate. Compared with WOD automatically modulating the size of irrelevant values, the strict selective rate of POP results in the poor performance on some data sets with high percentage of irrelevant values (i.e., data sets with low k), e.g., *SylvaP* and *SylvaA*. Note that the AUC value of POP on *Census* is very low, it is may because the initialisation of value subset selection mistakenly contains a large percentage of irrelevant values, and the value outlierness scoring deviates from the right path in the following iterations. CBRW models complicated but completed value couplings, which enables CBRW to obtain outstanding AUC performance compared with other four competitors. It is noteworthy that CBRW has poor performance on the data sets with high η_{noise} compared to WOD because full-space coupling learning used in CBRW is greatly influenced by these noisy features. In terms of ZERO and iForest, they work on feature subspace to detect outliers and are less sensitive to noisy features. LOF fails on most of the data sets. It is because the performance of LOF heavily depends on the choosing of neighbourhood size *MinPts* and is infected by the severe distance concentration effect caused by noisy features.

4.5 Scalability Test

We examine the scalability of WOD w.r.t both of data size and dimensionality. The data sets with varying dimensionality and data size are randomly downsampled from data set *CT* and *wap.wc*.

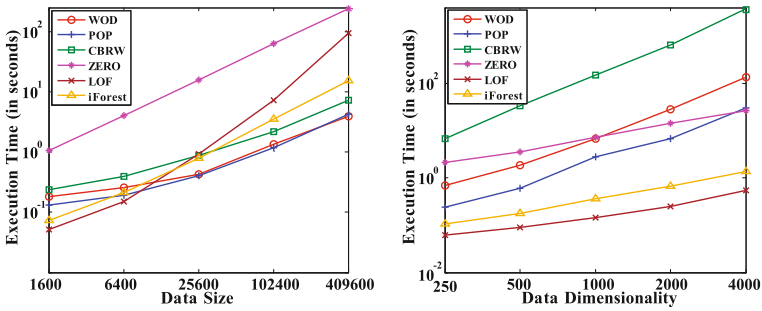


Fig. 2. Scalability Test w.r.t. Data Size and Dimensionality.

The scalability test results are presented in Fig. 2. As expected, WOD is linear w.r.t. data size and is quadratic w.r.t. the number of features. In the left panel, WOD runs comparably fast to POP, CBRW, and iForest. The result in the right

panel shows that WOD runs faster than CBRW and obtains comparably same execution time with POP and ZERO but runs slower than LOF and iForest with linear time complexity w.r.t. data dimensionality.

5 Conclusions

This paper introduces a novel outlier detector WOD that combines value clustering and weighted value coupling learning. Extensive empirical results show that WOD significantly outperforms five state-of-the-art outlier detectors by at least 6% AUC improvement on 12 real-world data sets. The experiment results validate the capability of employing value clustering to separate irrelevant values and illustrate the power of weighted value coupling learning to facilitate value outlierness estimation.

Acknowledgements. This work is supported by NSFC No.61472439, National Natural Science Foundation of China under Grant.

References

1. Aggarwal, C.C.: Outlier Analysis. Springer, New York (2017). https://doi.org/10.1007/978-1-4614-6396-2
2. Akoglu, L., Tong, H., Vreeken, J., Faloutsos, C.: Fast and reliable anomaly detection in categorical data. In: CIKM, pp. 415–424. ACM (2012)
3. Angiulli, F., Fassetti, F., Palopoli, L.: Detecting outlying properties of exceptional objects. ACM Trans. Database Syst. **34**(1), 7 (2009)
4. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech.: Theory Exp. **2008**(10), P10008 (2008)
5. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. ACM SIGMOD Rec. **29**(2), 93–104 (2000)
6. Campos, G.O., et al.: On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Min. Knowl. Discov. **30**(4), 891–927 (2016)
7. Cao, L.: Coupling learning of complex interactions. Inf. Process. Manag. **51**(2), 167–186 (2015)
8. Chen, Y., Dang, X., Peng, H., Bart, H.L.: Outlier detection with the kernelized spatial depth function. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 288–305 (2009)
9. He, Z., Xu, X., Huang, Z.J., Deng, S.: FP-outlier: frequent pattern based outlier detection. Comput. Sci. Inf. Syst. **2**(1), 103–118 (2005)
10. Li, J., et al.: Feature selection: A data perspective. CoRR abs/1601.07996 (2016)
11. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data **6**(1), 1–39 (2012)
12. Pang, G., Cao, L., Chen, L.: Outlier detection in complex categorical data by modelling the feature value couplings. In: IJCAI, pp. 1902–1908. AAAI Press (2016)
13. Pang, G., Cao, L., Chen, L., Liu, H.: Unsupervised feature selection for outlier detection by modelling hierarchical value-feature couplings. In: ICDM, pp. 410–419. IEEE (2016)

14. Pang, G., Cao, L., Chen, L., Liu, H.: Learning homophily couplings from non-IID data for joint feature selection and noise-resilient outlier detection. In: IJCAI, pp. 2585–2591. AAAI Press (2017)
15. Pang, G., Ting, K.M., Albrecht, D., Jin, H.: Zero++: harnessing the power of zero appearances to detect anomalies in large-scale data sets. J. Artif. Intell. Res. **57**, 593–620 (2016)
16. Pang, G., Xu, H., Cao, L., Zhao, W.: Selective value coupling learning for detecting outliers in high-dimensional categorical data. In: CIKM, pp. 807–816. ACM (2017)
17. Sathe, S., Aggarwal, C.C.: Subspace outlier detection in linear time with randomized hashing. In: ICDM, pp. 459–468. IEEE (2016)

Mining Local High Utility Itemsets

Philippe Fournier-Viger[1(✉)], Yimin Zhang[2], Jerry Chun-Wei Lin[3], Hamido Fujita[4], and Yun Sing Koh[5]

[1] School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen, China
philfv8@yahoo.com
[2] School of Computer Sciences and Technology, Harbin Institute of Technology, Shenzhen, China
mrzhangym@126.com
[3] Department of Computing, Mathematics and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
jerrylin@ieee.org
[4] Faculty of Software and Information Science, Iwate Prefectural University, Takizawa, Iwate, Japan
HFujita-799@acm.org
[5] Department of Computer Sciences, University of Auckland, Auckland, New Zealand
ykoh@cs.auckland.ac.nz

Abstract. High Utility Itemset Mining (HUIM) is the task of analyzing customer transactions to find the sets of items that yield a high utility (e.g. profit). A major limitation of traditional HUIM algorithms is that they do not consider that the utility of itemsets vary over time. Thus, traditional HUIM algorithms cannot find itemsets that have a high utility during specific time periods. This paper addresses this limitation by defining the problem of mining *local high utility itemsets* (LHUI). An efficient algorithms named LHUI-Miner is proposed to mine these patterns. Experimental results show that the proposed algorithms are highly-efficient and can find useful patterns not found by traditional HUIM algorithms.

Keywords: High-utility pattern mining · Local high-utility itemsets

1 Introduction

Frequent Itemset Mining (FIM) [1,2] is a fundamental data mining task, which consists of finding itemsets (sets of items) that are frequently purchased in a customer transaction database. However, it assumes that all items in a database are equally important and can appear at most once in each transaction. To address this limitation, *High-Utility Itemset Mining* (HUIM) [3–6] has recently emerged as an important data mining task. It consists of finding itemsets (sets of items) that yield a high utility (e.g. importance or profit) in customer transaction databases. An itemset is a High Utility Itemset (HUI) in a database if its utility is

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 450–460, 2018.
https://doi.org/10.1007/978-3-319-98812-2_41

no less than a user-specified minimum utility threshold. HUIM is widely viewed as a more difficult problem than FIM since the utility measure used in HUIM is not anti-monotonic, unlike the support measure used in FIM. In other words, the utility of an itemset may be greater, equal or smaller than the utility of its supersets. Thus, traditional FIM techniques cannot be directly used in HUIM to reduce the search space. HUIM algorithms such as Two-Phase [5] have thus introduced upper-bounds on the utility measure such as the TWU, which is anti-monotonic, to reduce the search space. Several more efficient algorithms have then been proposed, such as UP-Growth, HUI-Miner and FHM [3,4,6].

Though, HUIM has many applications such as click stream analysis, market basket analysis and biomedical applications [4,6], a major limitation of HUIM is that is that it ignores the time at which transactions were made. But considering the timestamps of transactions is important as the utility of patterns may vary over time. A few extensions of HUIM and FIM consider time. For example, *periodic high-utility itemset mining* [8] discovers itemsets that are periodically bought by customers and yield a high profit. However, it does not consider the timestamps of transactions (only their relative order), and tend to find patterns that are stable in terms of utility over long periods of time. Another related work is *High On-shelf Utility itemset Mining* (HOUM) [10], where each transaction is associated to a user-predefined time period (e.g. winter), and each item is associated to a set of periods indicating when it was sold. However, a major limitation of HOUM is that the utility of itemsets is calculated based on predefined time periods (e.g. *winter*), which is unrealistic because many products may have on-shelf time periods that may not match the predefined periods. Besides, HOUM also tend to find patterns that are stable in terms of utility in time periods where they are sold. Another related problem is to detect time points where the frequency of itemsets change significantly in data streams [7,9]. However, it also only consider the relative order of transactions instead of their real time stamps. In other words, it makes an unrealistic assumption that the time interval between any consecutive transactions is the same.

To find patterns that are profitable in non-predefined time periods, this paper proposes to discover a new type of patterns called *Local High Utility Itemsets* (LHUI). It consists of finding itemsets that yield a utility that is no less than a user-specified threshold during one or more time periods having a minimum time length. This allows to discover useful patterns such that the itemset $\{schoolbag, pen, notebook\}$ yields a high profit during the back-to-school shopping season, while not being a HUI in the whole database. An efficient algorithm called LHUI-Miner is designed to discover LHUIs. It relies on a novel data structure named LU-list.

The rest of this paper is organized as follows. Section 2 introduces preliminaries and defines the problems of mining LHUIs. Section 3 presents the proposed algorithm. Section 4 presents the experimental evaluation. Lastly, Sect. 5 draws the conclusion.

2 Preliminaries and Problem Statement

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items. A transaction T is a subset of items purchased by a customer ($T \subseteq I$). A *transaction database* is a set of transactions $D = \{T_1, T_2, \ldots, T_m\}$, where each transaction T_{tid} ($1 \leq tid \leq m$) has a unique identifier tid. Moreover, let $t(T)$ denotes the time at which a transaction T was made. Each item $i \in I$ is associated with a positive number $p(i)$, called its external utility, which indicates its relative importance (e.g. unit profit). For each transaction T and item $i \in T$, a positive number $q(i, T)$ is called the internal utility of i, and represents the purchase quantity of i in T. For example, consider the database of Table 1, which will be used as running example. This database contains eight transactions $(T_1, T_2, \ldots, T_8)$ and five items (a, b, c, d, e), where internal utilities (e.g. quantities) are shown as integers beside items. For instance, transaction T_1 indicates that 2, 2 and 1 units of items b, c and e were purchased, respectively. Table 2 indicates that the external utilities (unit profits) of these items are 2, 1 and 2. In this example, timestamps of transactions $T_1, T_2 \ldots T_8$ are $d_1, d_3, \ldots d_{10}$, representing days ($d_i = i$-th day). But other time units can be used such as milliseconds, and transactions can be simultaneous.

Table 1. A transaction database

Trans.	Items	Timestamp
T_1	$(b,2),(c,2),(e,1)$	d_1
T_2	$(b,4),(c,3),(d,2),(e,1)$	d_3
T_3	$(b,2),(c,2),(e,1)$	d_3
T_4	$(a,2),(b,10),(c,2),(d,10),(e,2)$	d_5
T_5	$(a,2),(c,6),(e,2)$	d_6
T_6	$(b,4),(c,3),(e,1)$	d_7
T_7	$(a,2),(c,2),(d,2)$	d_9
T_8	$(a,2),(c,6),(e,2)$	d_{10}

Table 2. External utilities of items

Item	a	b	c	d	e
Unit profit	5	2	1	2	3

Definition 1 (Utility of an item/itemset). The utility of an item i in a transaction T is defined as $u(i, T) = p(i) \times q(i, T)$. A set $X \subseteq I$ is an itemset. The utility of X in a transaction T is defined as $u(X, T) = \sum_{i \in X \wedge X \subseteq T} u(i, T)$. The utility of an itemset X in a database is defined as $u(X) = \sum_{T \in D \wedge X \subseteq T} u(X, T)$ [5].

For example, the utility of item b in T_1 is $u(b, T_1) = 2 \times 2 = 4$. The utility of itemset $\{b, c\}$ in T_1 is $u(\{b, c\}, T_1) = u(b, T_1) + u(c, T_1) = 2 \times 2 + 1 \times 2 = 6$. The utility of itemset $\{b, c\}$ in the database is $u(\{b, c\}) = u(\{b, c\}, T_1) + u(\{b, c\}, T_2) + u(\{b, c\}, T_4) + u(\{b, c\}, T_6) = 6 + 11 + 6 + 11 = 34$.

An itemset X is said to be a *high utility itemset* (HUI) if its utility $u(X)$ is no less than a user-specified positive threshold *minutil* [5]. The utility measure is not anti-monotonic. Thus pruning strategies used in FIM cannot be directly used in HUIM. To reduce the search space in HUIM, the Transaction Weighted Utilization (TWU) upper-bound was introduced [5].

Definition 2 (Transaction weighted utilization). The utility of a transaction T is defined as $tu(T) = \sum_{i \in T} u(i, T)$. The TWU of an itemset X is the sum of the utilities of transactions containing X, i.e. $TWU(X) = \sum_{X \subseteq T \wedge T \in D} tu(T)$.

Property 1. For two itemsets $X \subseteq X'$, $u(X') \leq TWU(X') \leq TWU(X)$ [5].

Thus, for an itemset X, $TWU(X)$ is an upper-bound on $u(X)$, and the TWU is anti-monotonic. Hence, if $TWU(X) < minutil$, itemset X and all its supersets can be pruned. A major limitation of HUIM is that it cannot find itemsets that yield a high utility in specific time periods but not in the whole database. For example, for $minutil = 50$, itemset $\{a, c\}$ is a HUI since $u(\{a, c\}) = 56 > 50$, and $\{d, e\}$ is not a HUI since $u(\{d, e\}) = 36 < minutil$. But from time $d5$ to $d6$, $\{d, e\}$ yields a utility that is more than twice the utility of $\{a, c\}$ (a utility of 26 instead of 12). Thus, during this period, $\{d, e\}$ is more interesting than $\{a, c\}$, but is ignored in HUIM. To address this problem, we propose a new type of patterns called *Local High utility itemsets* (LHUIs), defined as follows.

Definition 3 (Window). A window denoted as $W_{i,j}$ is the set of transactions from time i to j, i.e. $W_{i,j} = \{T | i \leq t(T) \leq j \wedge T \in D\}$, where i, j are integers. The length of a window $W_{i,j}$ is defined as $length(W_{i,j}) = j - i + 1$. The length of a database D containing m transactions is $W_D = t(T_m) - t(T_1) + 1$. A window $W_{k,l}$ is said to **subsume** another window $W_{i,j}$ iff $W_{i,j} \subsetneqq W_{k,l}$.

Definition 4 (Local high utility itemset). The utility of an itemset X in a window $W_{i,j}$ is defined as $u_{i,j}(X) = \sum_{T \in W_{i,j} \wedge X \subseteq T} u(X, T)$. An itemset X is a local high utility itemset (LHUI) if there exists a window $W_{i,j}$ such that $length(W_{i,j}) = minLength$ and $u_{i,j}(X) \geq lMinutil$, where $minLength \leq W_D$ and $lMinutil > 0$ are user-specified thresholds representing a minimum length and utility, respectively. Moreover, let $LHUIs$ denotes the set of all LHUIs.

For example, for $minLength = 3$ and $lMinutil = 20$, itemset $\{b, c\}$ is a LHUI since for the window W_{d_1,d_3} the utility of $\{b, c\}$ is $u_{d_1,d_3}(\{b, c\}) = u(\{b, c\}, T_1) + u(\{b, c\}, T_2) + u(\{b, c\}, T_3) = 6 + 11 + 6 = 23 \geq lMinutil = 20$, and $length(W_{d_1,d_3}) = 3 - 1 + 1 = 3 = minLength$.

Theorem 1 (Relationship between LHUIs and HUIs). *If* $minutil = lMinutil \times \lceil \frac{W_D}{minLength} \rceil$, *then* $HUIs \subseteq LHUIs$.

Proof. If Theorem 1 is false, then there exists an itemset $X \in HUIs$ such that $X \notin LHUIs$. This implies that for all window $W_{i,j}$ such that $length(W_{i,j}) = minLength, u_{i,j}(X) < lMinutil$. Assume that we divide the database into $\lceil \frac{W_D}{minLength} \rceil$ non overlapping windows having a length $minLength$ (note that there may be a window of length less than $minLength$ if $\frac{W_D}{minLength}$ is not an integer). Let WU_i denotes the utility of X in the i-th window. Since $\forall WU_i, WU_i < lMinutil$, it follows that $u(X) = \sum_{i=0}^{\lceil \frac{W_D}{minLength} \rceil} WU_i < lMinutil \times \lceil \frac{W_D}{minLength} \rceil = minutil$. There is a contradiction between $u(X) < minutil$ and $X \in HUIs$. Thus, the Theorem 1 is proven. □

Definition 5 (LHUI period). For an itemset X, a window $W_{i,j}$ is a *LHUI period* if for each window $W_{k,l} \subseteq W_{i,j}$ of length $minLength$, $u_{k,l}(X) \geq lMinutil$. A LHUI period $W_{i,j}$ is said to be a *maximum LHUI period* if there is no LHUI period $W_{o,p}$ such that $W_{i,j} \subset W_{o,p}$.

For example, consider that $minLength = 5$ and $lMinutil = 30$. The window W_{d_1,d_6} is a LHUI period of $\{a, b, c\}$, because W_{d_1,d_5} and W_{d_2,d_6} are its two sub-windows of length $minLength$, and $u_{d_1,d_5}(\{a, b, c\}) = u_{d_2,d_6}(\{a, b, c\}) = 32 \geq lMinutil$. The maximum LHUI period of itemset $\{a, b, c\}$ is W_{d_1,d_9}.

Definition 6 (Local High Utility Itemset mining). The problem of Local High Utility Itemset Mining (LHUIM) is to find all LHUIs and their maximum LHUI periods given the parameters $minLength$ and $lMinutil$.

For example, given the database of Table 1, $minLength = 5$ and $lMinutil = 30$, 25 LHUIs are found with their maximum LHUI periods, including $\{a, d\}$:$[d_5, d_9]$, $\{a, b, c\}$:$[d_5, d_5]$, $\{c, e\}$:$[d_3, d_7]$, $\{b, e\}$:$[d_1, d_7]$, $\{a, c, e\}$:$[d_5, d_{10}]$, $\{b, d, e\}$:$[d_3, d_5]$, $\{b, c, e\}$:$[d_1, d_7]$, $\{b, c, d, e\}$:$[d_3, d_5]$. For an itemset X, the numbers between brackets indicate the first and last timestamps of transactions in the LHUI period of X that contains the itemset. This notation is called the *abbreviated LHUI period of* X. For example, although the LHUI period of $\{a, b, c\}$ is $[d_1, d_9]$, it only appears in d_5. Thus, its LHUI period is denoted as $[d_5, d_5]$. For $minutil = lMinutil \times \lceil \frac{W_D}{minLength} \rceil = 60$, traditional HUIM algorithms find 5 HUIs: $\{b, e\}$, $\{a, c, e\}$, $\{b, d, e\}$, $\{b, c, d, e\}$ and $\{b, c, e\}$. Here, all HUIs are LHUIs, and these latter provides LHUI period information.

3 Proposed Algorithm

This section presents an algorithm to efficiently mine LHUIs, named LHUI-Miner, which extends the HUI-Miner [4] algorithm. The algorithm is designed to explore the search space of itemsets by following a total order $\succ$ on items in I. It is said that an itemset Y is an *extension* of an itemset X if $Y = X \cup \{j\} \wedge \forall i \in X,\ j \succ i$. The proposed algorithms utilize a novel data structure called *Local Utility-list* (LU-list) to store information about each itemset, which adapts the utility-list [4] structure to store period information. The algorithm first scan the database to create a LU-list for each item. Then, it explores the search space of itemsets using a depth-first search, by combining pairs of itemsets to generate their extensions and their LU-lists. A LU-list allow to determine if an itemset is a LHUI without scanning the database. The LU-list structure is defined as follows.

Let $\succ$ be any total order on I. The *LU-list* of an itemset X contains a tuple for each transaction that contains X. A *tuple* (also called *element*) has the form $(tid, iutil, rutil)$, where tid is the identifier of a transaction T_{tid} containing X, $iutil$ is the utility of X in T_{tid}. i.e. $u(X, T_{tid})$, and $rutil$ is defined as $\sum_{i \in T_{tid} \wedge \forall j \in X, i \succ j} u(i, T_{tid})$ [4]. Moreover, the LU-list contains two sets named $iutilPeriods$ and $utilPeriods$, which stores the abbreviated maximum LHUI periods and PLHUI periods of X, respectively. The PLHUI periods of X are the periods where X and its extensions could be LHUIs.

Definition 7 (PLHUI period). The remaining utility of an itemset X in a window $W_{k,l}$ is defined as $ru_{k,l}(X) = \sum_{i \in T \wedge T \in W_{k,l} \wedge X \subseteq T \wedge \forall j \in X, i \succ j} u(i,T)$. For an itemset X, a window $W_{k,l}$ is a *PLHUI period* (promising LHUI period) if for each window $W_{y,z} \subseteq W_{k,l}$ of length $minLength$, $u_{y,z}(X) + ru_{y,z}(X) \geq lMinutil$.

For example, consider that $a \prec b \prec c \prec d \prec e$, $minLength = 3$ *days* and $lMinutil = 22$. The LU-list of $\{a\}$ is $\{(T_4, 10, 48), (T_5, 10, 12), (T_7, 10, 6), (T_8, 10, 12)\}$, $iutilPeriods = \emptyset$, and $utilPeriods = \{[d_5, d_6], [d_9, d_{10}]\}$. The LU-list of $\{d\}$ is $\{(T_2, 4, 3), (T_4, 20, 6), (T_7, 4, 0)\}$, $iutilPeriods = \{[d_3, d_5]\}$, and $utilPeriods = \{[d_3, d_5]\}$. And the LU-list of $\{a, d\}$ is $\{(T_4, 30, 6), (T_7, 14, 0)\}$, $iutilPeriods = \{[d_5, d_5]\}$, and $utilPeriods = \{[d_5, d_5]\}$. The LU-list of an itemset provides useful information. If $iutilPeriods$ is not empty, then X is a LHUI. And if $utilPeriods$ is empty, then it can be proven that all its extensions and transitive extensions cannot be LHUIs or PHUIs and can be pruned from the search space. The proof of this property is omitted due to space limitation.

The LHUI-Miner Algorithm. LHUI-Miner takes as input a transaction database with utility values and the $lMinutil$ and $minLength$ thresholds. The algorithm first scan the database to calculate the TWU of each item. At the same time, an array $tid2time$ is constructed, where the i-th position stores the timestamp of transaction $t(T_i)$. Thereafter, the algorithm only consider items having a TWU no less than $lMinutil$, denoted as I^*. The TWU values of items are used to set a total order $\succ$ on I^*, which is the order of ascending TWU values [3]. A database scan is then performed to reorder items in each transaction according to $\succ$, and build the LU-list of each item $i \in I^*$. Then, the depth-first search of itemsets starts by calling the recursive *LHUI-Search* procedure with $\emptyset$, the LU-lists of 1-itemsets, $lMinutil$ and $minLength$.

LHUI-Search (Algorithm 1) takes as input (1) an itemset P, (2) a set of extensions of P, (3) $lMinutil$, and (4) $minLength$. The procedure then checks if $iutilPeriods$ is empty in the LU-list of each extension Px of P. If yes, Px is a LHUI and it is output with its abbreviated maximum LHUI periods (derived from $iutilPeriods$ and $tid2time$). Moreover, if $utilPeriods$ is not empty, it means that extensions of Px should be explored. This is performed by merging Px with each extension Py of P such that $y \succ x$ to form an extension of the form Pxy containing $|Px| + 1$ items. The LU-list of Pxy is then constructed using the *Construct* procedure of *HUI-Miner*, which join the tuples in the LU-lists of P, Px and Py. Thereafter, $iutilPeriods$ and $utilPeriods$ in the LU-list of Pxy are constructed by calling the *generatePeriods* procedure. Then, *LHUI-Search* is called with Pxy to calculate its utility and explore its extension(s) using a depth-first search. The *LHUI-Miner* procedure starts from single items, it recursively explores the search space of itemsets by appending single items and it only prunes the search space based on the properties of LU-list. It can be easily seen that this procedure is correct and complete to discover all LHUIs.

The *generatePeriods* procedure (Algorithm 2) takes as input (1) a LU-list lUl, (2) $lMinutil$ and (3) $minLength$. The procedure slides a window over lUl using two variable $winStart$ (initialized to 0; the first element of lUl), and $winEnd$. The procedure first scan lUl to find $winEnd$ (the end index of the

Algorithm 1. LHUI-Search

input : P: an itemset, $ExtensionsOfP$: extensions of P, $lMinutil$: a user-specified threshold, $minLength$: a window length threshold
output: the set of LHUIs and their abbreviated maximum LHUI periods

```
1  foreach itemset Px ∈ ExtensionsOfP do
2      if Px.LUList.iutilPeriods ≠ ∅ then output Px with Px.LUList.iutilPeriods;
3      if Px.LUList.utilPeriods ≠ ∅ then
4          ExtensionsOfPx ← ∅;
5          foreach itemset Py ∈ ExtensionsOfP such that y ≻ x do
6              Pxy.LUList ← Construct (P, Px, Py);
7              generatePeriods (Pxy, lMinutil, minLength);
8              ExtensionsOfPx ← ExtensionsOfPx ∪ Pxy;
9          end
10         LHUI-Miner (Px, ExtensionsOfPx, minutil, minLength);
11     end
12 end
```

first window), $iutils$ (sum of $iutil$ values in the first window) and $rutils$ (sum of $rutil$ values in the first window). Then, it repeats the following steps until the end index $winEnd$ reaches the last tuple of the LU-list: (1) increase the start index $winStart$ until the timestamp changes, and at the same time decrease $iutils$ ($rutils$) by the $iutil$ ($rutil$) values of tuples that exit the current window, (2) increase the end index until the window length is no less than $minLength$, and at same time increase $iutils$ ($rutils$) by the $iutil$ ($rutil$) values of tuples that enter the current window, (3) compare the resulting $iutils$ and $iutils + rutil$ values with $lMinutil$ to determine if the current period should be merged with the previous period or added to $iutilPeriods$ and $utilPeriods$ (line 14 to 15). Merging is performed to obtain the maximum LHUI and PLHUI periods.

Algorithm 2. The generatePeriods procedure

input : lUl: a LU-list, $lMinutil$: a user-specified utility threshold, $minLength$: a user-specified window length threshold

```
1  winStart = 0;
2  Find winEnd (the end index of the first window in ul), iutils (sum of iutil values of the
   first window), rutils (sum of rutils values of the first window);
3  while winEnd < lUl.size do
4      while ul.get(winStart).time is same as previous index do
5          iutils = iutils − lUl.get(winStart).iutil;
6          rutils = rutils − lUl.get(winStart).rutil;
7          winStart = winStart + 1;
8      end
9      while ul.get(winEnd).time ≤ ul.get(winStart).time + minLength do
10         iutils = iutils + lUl.get(winEnd).iutil;
11         rutils = rutils + lUl.get(winEnd).rutil;
12         winEnd = winEnd + 1;
13     end
14     merge the [winStart, winEnd] period with the previous period if iutils ≥ lMinutil.
       Otherwise, add it to lul.iutilPeriods;
15     merge the [winStart, winEnd] period with the previous period if iutils + rutils ≥
       lMinutil. Otherwise, add it to lul.utilPeriods;
16 end
```

Optimizations. To improve the performance of LHUI-Miner, the next paragraphs describe three optimizations.

Strategy 1. Discarding unpromising items using the sliding window. A modified version of Property 1 is used in LHUI-Miner to discard items that cannot be in a LHUI or PHUI. The TWU of an item i in a window $W_{k,l}$ is defined as $TWU_{k,l}(i) = \sum_{i \in T \wedge T \in W_{k,l}} tu(T)$. During the first database scan, if there is an item i such that for any window $W_{k,l}$ of length $minLength$, $TWU_{k,l}(i) < lMinUtil$, then item i is discarded.

Strategy 2. Discarding unpromising transactions using the sliding window. If for each item i in a transaction T, $TWU_{k,l}(i) < lMinutil$ for each window $W_{k,l}$ of length $minLength$ containing T, then the transaction T is discarded because this transaction cannot be in any LHUI period.

Strategy 3. Discarding unpromising tuples in each LU-list. The LU-list of an itemset X can stores numerous tuples that represents the transactions where X appears. However, some of those transactions are not in any PLHUI periods. Thus, these transactions are not in the LHUI periods of X and those of its transitive extensions. Hence, this strategy does not store the tuples representing these transactions in the LU-list of each itemset X. This reduces the runtime of the algorithms since performing the intersection of LU-lists and scanning LU-lists is faster for smaller LU-lists.

Strategy 1 and 2 are applied once, during the first database scan, while Strategy 3 is applied during LU-list construction. The proofs that these strategies are correct are ommitted due to space limitation. But they can be easily derived from the previous definitions.

4 Experimental Evaluation

Experiments were performed to assess the performance of LHUI-Miner on a computer having an Intel Xeon E3-1270 v5 processor running Windows 10, and 16 GB of free RAM. The performance of LHUI-Miner were compared with non-optimized versions and the HUI-Miner algorithm for mining HUIs. Four real-life datasets commonly used in the HUIM litterature were used: *mushroom*, *retail*, *kosarak* and *e-commerce*. They represent the main types of data typically encountered in real-life scenarios (dense, sparse, and long transactions). Let $|I|$, $|D|$ and A represents the number of distinct items, transactions and average transaction length. *mushroom* is a dense dataset ($|I|$ = 16,470, $|D|$ = 88,162, A = 23). *kosarak* is a dataset that contains many long transactions ($|I|$ = 41,270, $|D|$ = 990,000, A = 8.09). *retail* is a sparse dataset with many different items ($|I|$ = 16,470, $|D|$ = 88,162, A = 10,30). *e-commerce* is a real-world dataset ($|I|$ = 3,803, $|D|$ = 17,535, A = 15.4), containing customer transactions from 01/12/2010 to 09/12/2011 of an online store. For the other datasets, external utilities of items are generated between 1 and 1,000 using a log-normal distribution and quantities of items are generated randomly between 1 and 5, as in [4,6]. Besides, the timestamps of transactions in these three databases are generated

by adopting the same distribution as the e-commerce database. The source code of algorithms and datasets can be downloaded from the SPMF library [11].

LHUI-Miner was run with $minLength = 90$ *days* for *e-commerce* and 30 days for the other datasets. Thereafter, *lhui-op* denotes LHUI-Miner with optimizations; and *lhui-non-op* denotes LHUI-Miner without optimization. Algorithms were run on each dataset, while decreasing $lMinutil$ (for HUI-Miner $minutil = lMinutil \times \lceil \frac{W_D}{minLength} \rceil$) until they became too long to execute, ran out of memory or a clear trend was observed. Figure 1 compares the execution times of LHUI-Miner with and without optimization. Figure 2 compares the numbers of LHUIs and HUIs, respectively generated by these algorithms.

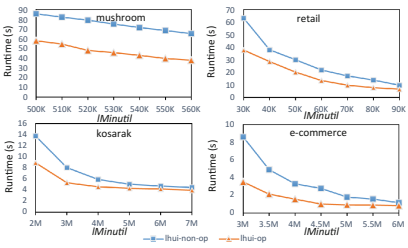


Fig. 1. Execution times

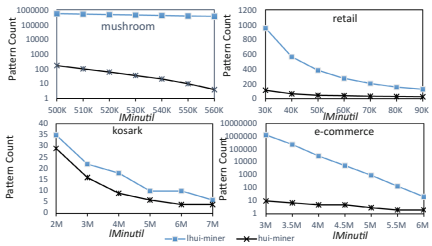


Fig. 2. Number of patterns found

It can be observed that in most cases, optimizations reduce the runtime. In some cases, the optimized algorithm is one time faster than the non-optimized algorithm, while in some cases there is little improvement. We also compared the execution time of HUI-Miner ($lMinutil \times \lceil \frac{W_D}{minLength} \rceil$) with the proposed algorithm. HUI-Miner is generally much faster because HUI-Miner generates much less patterns than LHUI-Miner (in other words, the problem of HUI mining is easier). However, when the number of patterns found by HUI-Miner is similar to the proposed algorithm, their runtimes are similar. Thus, because HUI-Miner is defined for a different problem, its results are not shown in Fig. 1.

A second observation is that the number of LHUIs is much more than the number of HUIs in most cases. This is reasonable since an itemset is much more likely to be high utility in at least one window than in the whole database. For example, on mushroom ($W_D = 180$ *days*), $minutil = 500,000$, $lMinutil = 83,333$, $minlength = 30$ *days*, there are 168 HUIs and 549,479 LHUIs. Among all patterns found, some interesting LHUIs are found in *e-commerce*. For instance, for $lMinutil = 1,432,360$ and $minlength = 90$ *days*, the itemset {pink polkadot bag, black and white baroque bag} has a PLHUI period from 2011/07/19 to 2011/11/02 where it generates a high utility, while the itemset is not a HUI in the whole database for $minutil = 6,000,000$. This information can be very useful for a retail store manager as it shows that this product generates a high profit from the end of July to early November. This can be useful to replenish stocks and offer promotions on this set of products during that period for the following year. Lastly, another observation is that for *kosarak*, the difference between the

number of LHUIs and HUIs is very small compared to other datasets. The reason is that the utilities of patterns do not vary much over time in *kosarak*.

5 Conclusion

To find itemsets that yield a high utility in non-predefined time periods and consider timestamps of transactions, this paper defined the problem of mining Local High-Utility Itemsets (LHUIs). An algorithm named LHUI-Miner (Local High-Utility Itemset Miner) was designed to efficiently discover LHUIs. Besides, three strategies were proposed to improve the performance of LHUI-Miner. An experimental evaluation has shown that LHUI-Miner can discover useful patterns that traditional HUIM could not find and that strategies reduces the runtime and memory consumption.

For future work, we will design concise representations of LHUIs to show a summary of all patterns to the user. Moreover, we will adapt the concept of this paper to other pattern mining problems such as sequential pattern mining and episode mining.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of 20th International Conference on Very Large Databases, pp. 487–499. Morgan Kaufmann, Santiago de Chile (1994)
2. Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng. **12**(3), 372–390 (2000)
3. Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V.S.: FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Andreasen, T., Christiansen, H., Cubero, J.-C., Raś, Z.W. (eds.) ISMIS 2014. LNCS (LNAI), vol. 8502, pp. 83–92. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08326-1_9
4. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, pp. 55–64. ACM, Maui (2012)
5. Liu, Y., Liao, W., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005). https://doi.org/10.1007/11430919_79
6. Tseng, V.S., Shie, B.-E., Wu, C.-W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. **25**(8), 1772–1786 (2013)
7. Wan, Q., An, A.: Discovering transitional patterns and their significant milestones in transaction databases. IEEE Trans. Knowl. Data Eng. **21**(12), 1692–1707 (2009)
8. Lin, J.C.W., Zhang, J., Fournier-Viger, P., Hong, T.P., Zhang, J.: A two-phase approach to mine short-period high-utility itemsets in transactional databases. Adv. Eng. Inform. **33**, 29–43 (2017)
9. Loglisci, C., Ceci, M., Malerba, D.: Relational mining for discovering changes in evolving networks. Neurocomputing **150**, 265–288 (2015)

10. Fournier-Viger, P., Zida, S.: FOSHU: faster on-shelf high utility itemset mining-with or without negative unit profit. In: Proceedings of 30th Annual ACM Symposium on Applied Computing, pp. 857–864. ACM, Salamanca (2015)
11. Fournier-Viger, P., et al.: The SPMF open-source data mining library version 2. In: Berendt, B., et al. (eds.) ECML PKDD 2016. LNCS (LNAI), vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8

Mining Trending High Utility Itemsets from Temporal Transaction Databases

Acquah Hackman[1], Yu Huang[2], and Vincent S. Tseng[2(✉)]

[1] Department of EECS-IGP, National Chiao Tung University, Hsinchu City, Taiwan, ROC
ahackman.cs03g@nctu.edu.tw

[2] Department of CS, National Chiao Tung University, Hsinchu City, Taiwan, ROC
yuvisu.cs04g@nctu.edu.tw, vtseng@cs.nctu.edu.tw

Abstract. In this paper, we address a novel and important topic in the area of HUI mining, named *Trending High Utility Itemset (TrendHUI)* mining, with the promise of expanding the applications of HUI mining with the power of trend analytics. We introduce formal definitions for *TrendHUI* mining and highlighted the importance of the *TrendHUI* output. Moreover, we develop two algorithms, *Two-Phase Trending High Utility Itemset (TP-THUI) miner* and *Two-Phase Trending High Utility Itemset Guided (TP-THUI-Guided) miner*. Both are two-phase algorithms that mine a complete set of TrendHUI. *TP-THUI-Guided miner* utilizes a remainder utility to calculate the temporal trend of a given itemset to reduce the search space effectively, such that the execution efficiency can be enhanced substantially. Through a series of experiments, using three different datasets, the proposed algorithms prove to be excellent for validity and efficiency. To the best of our knowledge, this is the first work addressing the promising topic on *Trending High Utility Itemset* mining, which is expected to facilitate numerous applications in data mining fields.

Keywords: High utility itemset · Utility pattern mining
Trend analysis · Data mining

1 Introduction

The problem of *High Utility Itemset (HUI) mining* [1] evolved from the most common task in data mining, the *Frequent Itemset Mining (FIM)* [2]. In HUI mining, not only does the outputs have to satisfy user requirements with regards to minimum confidence and support count but also the minimum utility.

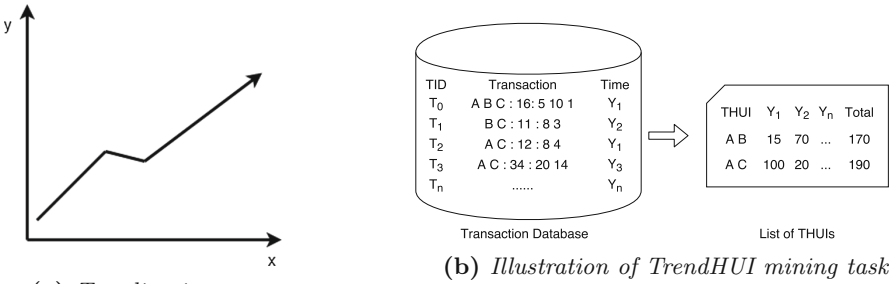
Most of the previous work [3–5] on HUI mining centred around developing efficient algorithms. Tseng et al. [6] used a tree structure in combination with the commonly used pruning strategy, TWU (Transaction Weighted Utility [3]), to develop the UP-Growth and UP-Growth+. EFIM [4] was the fastest algorithm at the time of writing this paper. In addition to using the TWU, EFIM also uses

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 461–470, 2018.
https://doi.org/10.1007/978-3-319-98812-2_42

two other upper bound strategies, the *local utility* and *sub-tree utility* for pruning to enhance execution time, and implements database projection and transaction merging to improve memory performance.

It is no doubt that HUI has numerous applications as already highlighted by previous research [3,4,6]. However, with the current representation and mining of HUI, which gives only a single value for a high utility itemset, it is almost impossible to make any predictions and forecasting for product sales in the retail industry. Single values of an itemset can only give information about the itemset's performance in the past.

In this paper, we address a novel and important topic in the area of HUI mining, named *Trending High Utility Itemset (TrendHUI)* mining. TrendHUI has the potential to extend the applications of HUI mining using promising technologies like time series mining, machine learning and other innovative artificial intelligence. Figure 1 illustrates the input and output of a TrendHUI mining task.



(a) *Trending itemset*

(b) *Illustration of TrendHUI mining task*

Fig. 1. Input and output of a TrendHUI miner

TrendHUI can be applied and expanded on different domains that are already benefiting from HUI mining. In market basket analysis, TrendHUI can be used to give more insightful information about items which are purchased together and their long-term effect. Applications in the stock market, e-commerce, and biomedicine will also be possible with TrendHUI mining. TrendHUI mining can additionally be used for recommendation systems to identify what itemsets are often purchased together, more importantly, highlighting their trend over time. The primary motivations for this research direction seek to rectify some of the earlier assumptions made in HUI mining to expand the research domain. These assumptions are as follows: (i) Current HUI algorithms ignore time concept in mining. In the real world, transaction database has a timeline that needs to be considered when mining. Also, (ii) Non-informative HUI mining output.

We have develop two algorithms, *Two-Phase Trending High Utility Itemset (TP-THUI) miner* and *Two-Phase Trending High Utility Itemset Guided (TP-THUI-Guided) miner* to approach this problem. To the best of our knowledge, this paper is the first work that addresses the interesting and important research

topic of mining *Trending High Utility Itemset (TrendHUI)*. The major contributions of this paper include (i) The new concept of TrendHUI being introduced formally which broadens the research horizon of HUI mining, (ii) Informative representation of HUI mining task, (iii) Novel algorithms are proposed for mining a complete set of TrendHUI efficiently, and (iv) Extensive experiments were conducted to show the feasibility of the proposed ideas and the performance of the designed algorithms.

2 Related Work

Almost all HUI algorithms can be placed in one of the two groups, two-phase and one-phase algorithm. The one-phase algorithms emerged after the Two-Phase [5], and they performed better than the former. Algorithms such as Two-Phase, BAHUI [7], UP-Growth and UP-Growth+ [6] are all examples of two-phase algorithms that are based on the TWU(Transaction Weighted Utilization) to reduce their search space. The downside of the two-phase paradigm was the problem of large irrelevant candidate generation [5], this problem was addressed using new algorithms which were engineered to avoid the candidate generation entirely. Using a single phase was a game changer speeding up execution time in 10 and 100 folds in both d^2HUP [8] and HUI-Miner [9]. Variants of HUI-Miner, also known as HUP-Miner [10] sought to improve on execution time and better performance. To improve the execution time, it is crucial to reduce the number of database scan as this slows down the mining process. By this point, two factors needed to be considered to improve the execution time of HUI mining are (i) candidate generation and (ii) Database scans, with the former becoming less of a concern as single phase algorithms skip this mining step. This is evident in a single phase state-of-the-art algorithm [4] which performs well in both memory management and execution time.

To understand the trend analysis in our research, we will first introduce some pioneering work in mining HUI in Streaming Data [11,12]. In data stream mining, time and sequence order are both essential components. Algorithms in mining high utility itemsets over stream data uses two fundamental paradigms: a time-fading [12] and a sliding window [11] paradigms.

3 Problem Definitions

HUI mining has already been around for a while now with some common definitions [6]. We will, therefore, introduce some definitions that are vital for the foundation of our research in TrendHUI.

3.1 Preliminaries

Let's represent a set of items $I = \{i_2, i_2, \ldots, i_m\}$, each item $i_j (1 \leqslant j \leqslant m)$ has a unit profit $pr(i_{jy})$ for each given year y, this is also referred to as the external

Utility. A $k-itemset X_y = \{i_1, i_2, \ldots, i_k\}$ is a set of distinct items with a length of k generated in y time frame. A transaction database $D = \{T_{1y}, T_{2y}, \ldots, T_{ny}\}$ contains a set of transactions, and each transaction $T_d (1 \leqslant d \leqslant n)$ has both a unique identifier d, called TID and a timestamp (time frame) y, at which the transaction occurred. In a given transaction, each item has a utility associated with it, and it is also referred to as the internal utility represented as $q(i, T_d)$.

For our running example, we consider the database in Table 1. It has three columns: unique transaction (TID), *Transaction*, and T for timestamps. Several transactions can belong to the same timestamp if we consider the time stamp as a time frame in the form of years or months. Table 1 contains five transactions (T_1 to T_5). T_3 contains items a and c, note that the timestamp is Y_1 whereas T_5 containing the same items has a timestamp of Y_3. Table 2 shows the external utilities of each item over time.

Table 1. Transaction database

TID	Transaction	T(Time)
1	(a,1)(c,2)(d,1)	Y_1
2	(a,1)(b,1)(c,8)	Y_1
3	(a,1)(c,4)	Y_2
4	(a,1)(b,1)(c,12)	Y_2
5	(a,1)(c,4)	Y_3

Table 2. External utility values per year

Item	Y_1	Y_2	Y_3
a	6	5	8
b	8	9	6
c	10	7	5
d	5	5	7

3.2 Definitions

Definition 1. *Time bound utility of an itemset*

The time bound utility of an itemset for a given time frame y in a given transaction T_d is denoted as $u(X_y, T_d)$ and defined as $u(X_y, T_d) = \sum_{i \in X_y} u(i_y, T_d)$ if $X_y \subseteq T_d$. The combine time utility of an itemset can be considered as the total utility of an itemset in a given database denoted as $U_{\forall y}(X)$ and defined as $U_{\forall y}(X) = \sum_{T_d \in g(X)} u(X_y, T_d)$.

Definition 2. *Utility Set*

The utility set of an itemset in a database D denoted as $U_{set}(X)$ and it is defined as $U_{set}(X) = \{u(X_{y_1}), u(X_{y_2}), \ldots, u(X_{y_n})\}$.

Definition 3. *The slope of a Utility Set*

The slope of a utility set is denoted as $s(U_{set}(\mathrm{X}))$ and it's defined as $s(U_{set}(\mathrm{X})) = \frac{\sum (x-\overline{x})(y-\overline{y})}{\sum (x-\overline{x})^2}$, where $x \in U_{set}(\mathrm{X})$, y is the time frame from Y_1 to Y_n.

Definition 4. *Remainder Utility*

According to Definitions 1 and 3, given an itemset X from list of HUI, the remainder utility is the total minus the sum of all utilities in the utility list of X. We denote the remainder utility of itemset X as $rU(X)$and it is defined as $rU(X) = u(X) - \sum (U_{set}(X))$.

Definition 5. *Trend Direction*

The trend direction δ is a user-defined value where δ may have the value of positive$(+)$ sign if the utility set's slope is greater than or equal to 0, or negative$(-)$ sign if the slope value is less than 0.

Definition 6. *Trending high utility itemset (TrendHUI)*

An itemset X is a Trending High Utility Itemset iff (i) $u(X) \geqslant minUtil$ and (ii) $s(U_{set}(X))$has the same direction as δ.

Problem Statement for TrendHUI Mining. Given a user specified minimum utility threshold *minUtil*, user specified trend direction δ, and a transaction database D which contains a set of transactions T with timestamps, the problem of *TrendHUI* mining is to identify all itemsets in D which has total utility no less than *minUtil* and has a slope direction equal to δ.

4 Proposed Methods

To deal with the targeted TrendHUI mining problem, we propose two methods, namely TP-THUI and TP-THUIGuided. The overall constructs are to mine the entire transaction database, then search the different time frames in the dataset for the values of the HUIs obtained. In phase one we mine the high utility itemset [4], then in phase two we search for the utilities of the HUI over the given time frame.

4.1 Two-Phase Trending High Utility Itemset (TP-THUI) Miner

Algorithm 1. Two-Phase Trending High Utility Itemset mining

```
1  function TP-THUI (D, H, Y, δ);
   Input  : D: transaction database, H: a set of HUI, Y: time frame from Y_1 to Y_n, δ: direction
   Output: complete set of TrendHUI
2  Initialize Ω ← ∅;
3  foreach high-utility itemset α ∈ H do
4      U_set({α}) ← ∅;
5      foreach time frame y ∈ Y do
6          U_set ← U_set ∪ u({α}_y) ;          /* Scan D to calculate u({α}_y) */
7      end
8      if slope(U_set({α})) satisfies δ then
9          Ω ← Ω ∪ {α};
10     end
11 end
12 return Ω;
```

The goal of phase one of the TP-THUI algorithm is to discover a complete set of HUIs as described above. In phase two, we obtain the list of HUI as the candidate list from phase one. Given time frames Y_1 to Y_n, the utility of each X needs to be obtained from the original database for each Y. See Algorithm 1 for details. Once these values are obtained, line 8 uses the slope function (see Definition 3) and determines if according to Definition 6 the given itemset is a THUI based on user specified trend direction.

The most expensive task using this approach is the database scan which tends to be quite expensive depending on the size of D and the number of HUIs obtained from phase one. A tremendous amount of time is wasted on line 6 for the negation of the desired pattern only to be discarded on line 8.

4.2 Two-Phase Trending High Utility Itemset Guided (TP-THUI-Guided) Miner

Algorithm 2. Two-Phase Trending High Utility Itemset Guided mining

```
   Input  : D: transaction database, H: a set of HUI, Y: time frame from Y_1 to Y_n, k: seed, δ:
            direction
   Output: complete set of TrendHUI
 1 Function TP-THUI-Guided(D, H, Y, k, δ):
 2     Initialize s ← 0 , t ← k − 1
 3     if δ is negative then
 4         t ← 0
 5         s ← k − 1
 6     return Miner(s, t, δ)
 7
 8 Function Miner(s, t, δ):
 9     Initialize Ω ← Ø
10     foreach high-utility itemset α ∈ H do
11         U_set({α}) ←array[Y_1...Y_n]
12         for time frame y ← s to t do
13             U_set[y] ← u({α}_y)                    /* Scan D to calculate u({α}_y) */
14         end
15         for time frame y ← k to Y_t do
16             U_set[y] ← u({α}_y)
17             d(α) ← slope(U_set({α}))
18             U_set[Y_t] ← u(α) − Σ_{y∈Y} u({α}_y)          /* Definition6 */
19             df(α) ← slope(U_set({α}))
20             U_set[Y_t] ← 0
21             if d(α) and df(α) is same direction then
22                 if d(α) satisfies δ then
23                     Ω ← Ω ∪ {α}
24         end
25     end
26     return Ω
```

We propose a second approach, TP-THUI-Guided, to reduce unnecessary database scans. The property we explored in this approach is that in a given $U_{set}(X)$, the positions of larger utility values are crucial in determining the trend direction of the TrendHUI. But prior to getting the slope value, we already know the total utility (actual utility) of all HU-Itemsets. Combining our knowledge of the actual utility value of an itemset we can incrementally determine the slope of the itemset using the remainder utility in Definition 4 to determine the temporal slope before calculating the actual slope.

The goal of this approach is to avoid obtaining the complete utility set of the negation of THUIs. This is achieved by first obtaining the first three items (*seed*) of the utility set for the given HUI. The seed is the initial utility set values that should be obtained. For positively trending itemsets, the seed and direction of the processing start at position 0 of the utility set and vice versa for negatively trending itemsets. Two influential positions in the utility set are then defined; (1) most positive and (2) most negative. By positioning the $rU(X)$ from Definition 4

on either of these extreme positions, the direction (positive or negative) of X is determined if the $rU(X)$ has no effect on the previously determined direction from the *seed*. The *seed* is then grown whenever we can't determine the direction of the temporal slope and lines 19 through 24 is then repeated until the direction of the itemset is determined. Thus, the algorithm can quickly discard the negated slope utility sets. The algorithm terminates when all items in the HUI list are scanned to mine the complete set of TrendHUIs. See Algorithm 2 for the details.

5 Experimental Evaluations

5.1 Experimental Setup and Dataset

Our experimental setup is designed to evaluate the algorithms' efficiency, correctness and the capability in obtaining complete set of TrendHUI from a given database. Our experiments were carried out on a 64 bit Windows 10 platform with 32 GB ram and an i5-4590 intel processor. All algorithms are implemented using the Java programming language. Our datasets are three (Chess, Accidents, and Mushroom) of the benchmark datasets of HUI mining [4].

The key parameter in measuring performance in our experimental setup is execution time. The experiment compares the performance of the TP-THUI algorithm with the TP-THUI-Guided algorithm both of which were developed for TrendHUI mining. We will observe their performances for mining both positively trending itemsets as well as negatively trending itemsets. Given a transactional database, the sum of positively trending and negatively trending itemsets must be equal to the number of HUI mined from the same database. Since these benchmark datasets do not have timestamps assigned to them, we have to assign timestamps to them as shown in Table 1.

5.2 Evaluation of TP-THUI and TP-THUI-Guided

We used the algorithms to separately mine positively trending itemsets and also negatively trending itemsets on the same dataset for comparison. Figure 2 shows the performance of both algorithms on the five-year range datasets using the Chess dataset. In this dataset, the TP-THUI-Guided algorithm thrived with definite improvement over the baseline algorithm. This improvement is expected because the outputs from these datasets are fairly balanced between negatively trending HUIs and positively trending HUIs. Which is to say, if we were mining positively trending itemsets, TP-THUI-Guided focuses on finding and avoiding the negatively trending itemsets. This makes it perform better than the baseline provided there are a good number of the undesired trends but performs slightly bad if there are a good number of the desired trends. This phenomenon explains the case of the Mushroom dataset.

The longer the time frame, the better the performance of TP-THUI-Guided algorithm since a great percentage of the utility set would have been avoided if they are unwanted itemsets. This can be seen in the the negative trend mining

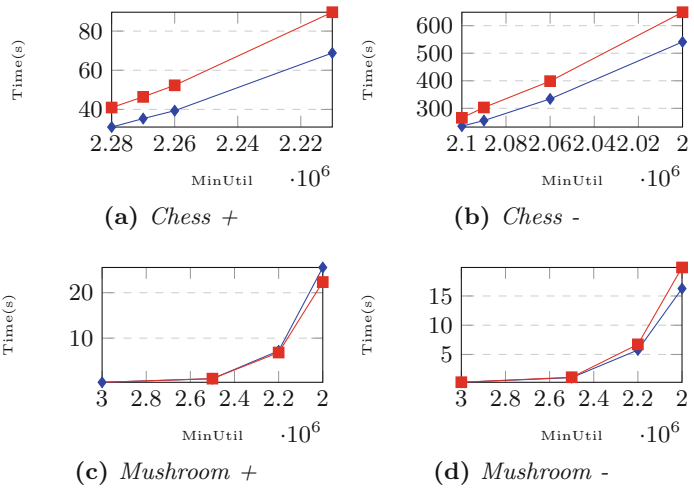


Fig. 2. The execution time of TP-THUI and TP-THUI-Guided using five-year period.

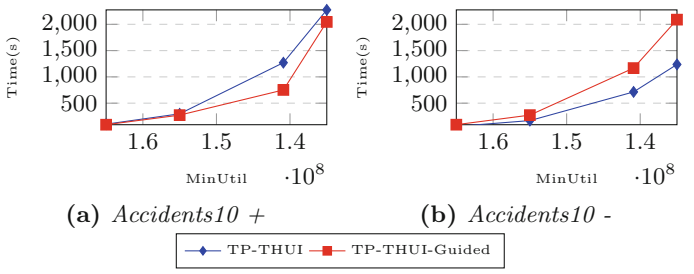


Fig. 3. The execution time of TP-THUI and TP-THUI-Guided using Accident10.

for the Accident dataset, the performance gain ranges from 18% to 38% in the 5 year period, whereas the gain is between 36% and 41% for the 10 year period see Fig. 3.

It is evident that for both algorithms, the execution time will increase significantly if the minimum utility value is set smaller. Also, the execution time depends on the number of HUIs obtained from phase one.

6 Conclusion

In this research work, we have developed two methods for the mining of TrendHUI, which is to mine a complete set of trending high utility itemsets (TrendHUI). We also represent the output of the TrendHUIs in an informative format. Moreover, we have proven that the mining of a complete set of TrendHUI is achievable in a feasible execution time, and most importantly we have presented

the first work on trend analysis in high utility itemset mining. This is important because the output of TrendHUI mining has many potential applications to be explored. For instance, using Time Series technologies, we should be able to point out the utility of an itemset in the next n time frames given previously known. This will also introduce other exciting topics that will change the current direction on the "fastest algorithm" race to an application and integration based research in the HUI mining domain. For the future work, we plan to work on datasets that can help us highlight the applications of high utility mining in a given domain such as stock market, biomedicine, and market basket analysis. We will also focus on single phase algorithms for TrendHUI mining.

Acknowledgements. This research was partially supported by Ministry of Science and Technology, Taiwan, under grant no. MOST 106-3114-E-009-008 and MOST 104-2221-E-009-128-MY3.

References

1. Chan, R., Yang, Q., Shen, Y.D.: Mining high utility itemsets. In: Third IEEE International Conference on Data Mining, ICDM 2003, pp. 19–26. IEEE (2003)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD Record, vol. 22, pp. 207–216. ACM (1993)
3. Liu, Y., Liao, W.K., Choudhary, A.: A fast high utility itemsets mining algorithm. In: Proceedings of the 1st International Workshop on Utility-Based Data Mining, pp. 90–99. ACM (2005)
4. Zida, S., Fournier-Viger, P., Lin, J.C.W., Wu, C.W., Tseng, V.S.: EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Knowl. Inf. Syst. **51**(2), 595–625 (2017)
5. Erwin, A., Gopalan, R.P., Achuthan, N.: CTU-Mine: an efficient high utility itemset mining algorithm using the pattern growth approach. In: 7th IEEE International Conference on Computer and Information Technology, CIT 2007, pp. 71–76. IEEE (2007)
6. Tseng, V.S., Shie, B.E., Wu, C.W., Philip, S.Y.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. **25**(8), 1772–1786 (2013)
7. Song, W., Liu, Y., Li, J.: BAHUI: fast and memory efficient mining of high utility itemsets based on bitmap. Int. J. Data Warehous. Min. (IJDWM) **10**(1), 1–15 (2014)
8. Liu, J., Wang, K., Fung, B.C.: Direct discovery of high utility itemsets without candidate generation. In: 2012 IEEE 12th International Conference on Data Mining (ICDM), pp. 984–989. IEEE (2012)
9. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, pp. 55–64. ACM (2012)
10. Krishnamoorthy, S.: Pruning strategies for mining high utility itemsets. Expert Syst. Appl. **42**(5), 2371–2381 (2015)
11. Shie, B.E., Philip, S.Y., Tseng, V.S.: Efficient algorithms for mining maximal high utility itemsets from data streams with different models. Expert Syst. Appl. **39**(17), 12947–12960 (2012)

12. Manike, C., Om, H.: Time-fading based high utility pattern mining from uncertain data streams. In: Kumar Kundu, M., Mohapatra, D.P., Konar, A., Chakraborty, A. (eds.) Advanced Computing, Networking and Informatics-Volume 1. SIST, vol. 27, pp. 529–536. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07353-8_61

Social Media vs. News Media: Analyzing Real-World Events from Different Perspectives

Liqiang Wang[1], Ziyu Guo[1], Yafang Wang[1(✉)], Zeyuan Cui[1], Shijun Liu[1], and Gerard de Melo[2]

[1] Shandong University, Jinan, China
yafang.wang@sdu.edu.cn
[2] Rutgers University, New Brunswick, USA

Abstract. For a long time, the news media has played a crucial role not only as an information provider, but also as an influential source of opinion and commentary. Nowadays, platforms such as Twitter provide an alternative to the traditional one-way interaction, enabling users to voice their opinions. Hence, one can obtain a more comprehensive picture of the range of perspectives on real-world events by considering both news and social media sources. In this paper, we compare mainstream news and Twitter data on 18 well-known real-world events from six different categories. We propose the event-based authoring model (EvA), a novel probabilistic model to capture the content characteristics of an event with respect to aspect, category and background word distributions. These results allow us to analyze the real-world events in different perspectives.

Keywords: News media · Social media · Real-world event analysis
Topic model

1 Introduction

As online social media and online news continue to mature, increasing numbers of people rely on online media platforms to obtain information about the world as well as to express their personal opinions about various kinds of events. Such platforms are now among the primary sources that people rely on to keep track of current events in the world. Hence, online media possess unprecedented power to influence people's opinions. Clearly, there are substantial differences between social media and news media with regard to linguistic properties, distributions of opinions, sentiment, subjectivity, authenticity, immediacy, to name but a few. One study [4] determined that the value of news has remained constant, but that most raw content now is available both to journalists and to social media users. However, with the increased prominence of social media platforms, these

L. Wang and Z. Guo—Contributed equally.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 471–479, 2018.
https://doi.org/10.1007/978-3-319-98812-2_43

now are also beginning to serve as gatekeepers on the news media. It has now become crucial to consider the interplay between social and news media, given the role that the two play in shaping the public's opinions on world events. Despite the abundance of research studying various aspects of online social media and of online news, this connection between the two has not received sufficient attention.

As more and more people participate in online discussions, analysts pay increasingly focus on mining the public perception, opinions, and online interactions pertaining to various real-world events, including, for instance, political events [8] and natural hazards [14]. One study [7] analyzed the engagement of Twitter users in response to real-world events. Although there have been numerous studies related to real-world event analytics, most of them focus on analyzing the behavior or online users rather than taking an event-centric perspective. To address this gap, the present study proposes a novel analytical model, considering event aspects as well as categories.

Both news and social media serve as vehicles for information authoring, dissemination, and diffusion. While there have been studies that sought to characterize specific aspects of how social and news media differ [11,15], in this paper, we attempt to provide a more comprehensive data-driven analysis of how news and social media differ, focusing on an event-centric perspective by highlighting how the two differ in covering the same set of events.

Our main contributions include:

1. We propose a new means of analyzing real-world events by accounting for both news and social media.
2. We introduce a new model called EvA to probabilistically model the events based on intuitions of how a journalist or user may compose an article.
3. We perform an analysis of event aspects to provide a contrastive analysis discriminating between news and social media.

2 Related Work

Event Detection. In this paper, we consider the somewhat rigid method of detecting real-world events based on keywords to compile our dataset. However, there are also several works that focus on extracting events using more involved methods. One study aims at related event discovery by extracting local events from web pages [10]. Certain previous work has also specifically aimed at detecting events in tweets [1]. For further references, the reader is referred to a recent survey that summarizes techniques for event detection in Twitter [2].

Event Analysis. In terms of data analysis, there has been previous work focusing on news and social media. ET-LDA [9] is a joint topic model for aligning events and corresponding feedback on Twitter. For news media, a recent method ranks the daily news events according to their importance [13]. Castillo et al. predict the life cycle of online news stories by analyzing reactions to the news on

social media [3]. For further information on the challenges and possible solutions for event analytics on social media, the reader is referred to a recent overview [6].

3 The EvA Model for Multi-perspective Event Analytics

3.1 Preliminaries and Definitions

To better understand the analytics and the model, we highlight some definitions in our model. **Events** refer to happenings in the world, e.g., the *Oscars*, *Nobel Prize*, or *terrorist attacks in France*. **An aspect** refers to a particular issue or subject that can be discussed about a given event. For instance, for our Academy Award event data, *Best picture*, *La La Land*, and *Emma Stone* are among the most trending aspects being discussed. **An event category** is a classification of events with regard to the event context, type, or attributes. For instance, based on the event context, we can consider categories such as *disasters*, *business*, *political events*, etc. **A background word** is an auxiliary word used by the authors to help in phrasing their thoughts, but typically does not express a specific opinion in the context under consideration, e.g., *actor*, *immigrate*, and *black*. In addition, the main notational conventions are enumerated in Table 1.

Table 1. The key notations used in this paper.

Notations	Description
E, D, T, C, N, B	Number of events ($e = 1 : E$), documents ($d = 1 : D$), aspects ($k = 1 : K$), event categories ($c = 1 : C$), words ($n = 1 : N$), and background word distribution
w, z, c	Word, aspect, category
x^0, x^1	Word indicators, one per word
ψ^0, ψ^1	Word bias on event category, aspect and background
$\lambda_0^0, \lambda_1^0, \lambda_0^1, \lambda_1^1$	Beta prior for hidden variables
$\theta, \phi, \sigma, \Omega$	Distribution of document-aspect, aspect-word, category-word and background-word
$\alpha, \beta, \delta, \pi$	Dirichlet prior for hidden variables

3.2 EvA Model Description

To address the aforementioned aims, we devise the EvA (Event-based Authoring) model in Fig. 1, inspired by the process of how a person – e.g., a journalist or blogger, or, alternatively, a social media user – authors a document, i.e., an article or a posting. Typically, the authoring process would be triggered by an event. We assume that the first decision that authors make is to settle on what event aspects they wish to write about. Because the purpose of writing a document is to express an opinion or to report on some aspect. Next, we assume that the

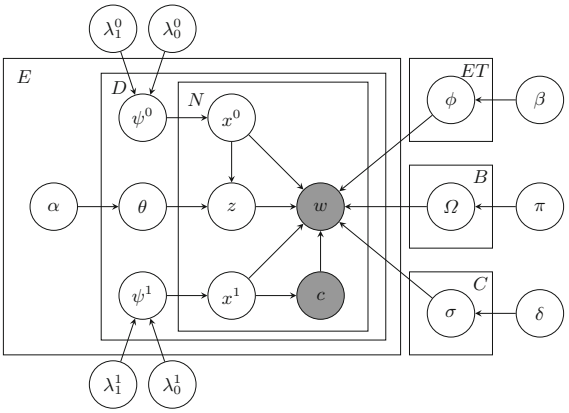


Fig. 1. EvA probabilistic graphical model

event category will exert special influence. For different category events, authors tend to follow different conventions while writing an article. Finally, authors will need to turn relevant information and thoughts on the various aspects of the event into a coherent narrative by organizing the words to form a suitable series of written sentences. For this, we distinguish three kinds of words: aspect words, category words, and background words. Background words, as mentioned, refer to auxiliary words, which can be thought of as being added last, as they merely assist in casting the thoughts and opinions into proper phrases.

Algorithm 1. The generation process for documents.

Draw background word distribution $\Omega \sim Dir(\pi)$
for each category $c = 1, ..., C$ **do**
  Draw category word multinomial distribution $\sigma_c \sim Dir(\delta)$
for each event $e = 1, ..., E$ **do**
  for each aspect $k = 1, ..., K$ **do**
    Draw aspect word multinomial distribution $\phi_{ek} \sim Dir(\beta)$
  for each document $d = 1, ..., D$ **do**
    Draw a Bernoulli distribution $\psi_d^0 \sim Beta(\lambda_1^0, \lambda_0^0)$
    Draw a Bernoulli distribution $\psi_d^1 \sim Beta(\lambda_1^1, \lambda_0^1)$
    Draw document aspect mixture $\theta_d \mid \alpha_e \sim Dir(\alpha_e)$
    for each word $n = 1, ..., N$ **do**
      Sample $x_n^0 \sim Bernoulli(\psi_d^0)$
      if $x_n^0 = 1$ **then**
        Sample an aspect $z_n \mid \theta_d \sim Multi(\theta_d)$
        Draw word $w_n \mid z_n \sim Multi(\phi_{ez_n})$
      if $x_n^0 = 0$ **then**
        Sample $x_n^1 \sim Bernoulli(\psi_{d_n}^1)$
        if $x_n^1 = 1$ **then**
          Draw word $w_n \mid c \sim Multi(\sigma_c)$
        if $x_n^1 = 0$ **then**
          Draw word $w_n \sim Multi(\Omega)$

3.3 The Inference Process

The posterior probability of the EvA model is as follows:

$$P(\theta_{1:D}, \phi_{1:K}, \sigma_{1:C}, \Omega, \psi^{\mathbf{0}}{}_{1:C}, \psi^{\mathbf{1}}{}_{1:K}, \mathbf{z}, \mathbf{x^0}, \mathbf{x^1} \mid \alpha, \beta, \delta, \pi, \lambda_0^0, \lambda_1^0, \lambda_0^1, \lambda_1^1, \mathbf{w}, \mathbf{c}) \quad (1)$$

Unfortunately, computing this posterior probability is intractable with all the variables. However, we can approximate it via collapsed Gibbs sampling. This involves integrating out the following hidden variables: $\theta_d, \phi_k, \sigma_c, \Omega, \psi_c^0$, and ψ_k^1.

We define C_{eq}^{EQ} as the number of times instance e appeared with instance q, e.g., C_{wk}^{WK} gives the number of times word w was assigned to aspect k. In addition, we use subscript $-i$ to denote the counting variable that excludes the i^{th} word index in the corpus. Moreover, $C_{\cdot w}^{BW}$ refers to the number of times that word w is sampled from the background word distribution. As a result, we finally obtain the following conditional posterior distribution:

$$P(x_n^0 = 1, z_n = k \mid w_n = w) \propto (C_{d1,-n}^{DX^0} + \lambda_1^0) \times \frac{C_{kw,-n}^{KW} + \beta}{\sum_{w'}(C_{kw',-n}^{KW} + \beta)} \times \frac{C_{dk,-n}^{DK} + \alpha_{ek}}{\sum_{k'}(C_{dk',-n}^{DK} + \alpha_{ek})} \quad (2)$$

$$P(x_n^0 = 0, x_n^1 = 1 \mid w_n = w, c_d = c) \propto (C_{d0,-n}^{DX^0} + \lambda_0^0) \times \frac{C_{d1,-n}^{DX^1} + \lambda_1^1}{C_{d1,-n}^{DX^1} + C_{d0,-n}^{DX^1} + \lambda_0^1 + \lambda_1^1} \times \frac{C_{cw,-n}^{CW} + \delta}{\sum_{w'}(C_{cw',-n}^{CW} + \delta)} \quad (3)$$

$$P(x_n^0 = 0, x_n^1 = 0 \mid w_n = w) \propto (C_{d0,-n}^{DX^0} + \lambda_0^0) \times \frac{C_{d0,-n}^{DX^1} + \lambda_0^1}{C_{d1,-n}^{DX^1} + C_{d0,-n}^{DX^1} + \lambda_0^1 + \lambda_1^1} \times \frac{C_{\cdot w,-n}^{BW} + \pi}{\sum_{w'}(C_{\cdot w',-n}^{BW} + \pi)} \quad (4)$$

α_e is a non-uniform vector related to the event e. Considering its simplicity and speed, we update α_{ek} according to $\alpha_{ek} = \frac{1}{N_e}\sum_d \frac{C_{dk}^{DK}}{C_{d\cdot}^{DK}}$ in each iteration of Gibbs sampling [12]. Assume that N_e is the number of documents in event e and document d belongs to event e.

4 Experiments and Analytics

4.1 Data Description

Our dataset includes 6 categories, and for each category we have 3 events. For each of these events, we compare the data from news and social media. The news media documents are sourced from the STICS project [5]. We rely on event keywords to filter these news articles so as to obtain related articles for a given event. The social media data is crawled from Twitter by relying on the Twitter API, using the same selection criteria. The overall statistics of the resulting dataset are given in Table 2.

Table 2. Dataset description

Category	ID	Keyword	Data period	Twitter	News
Armed conflicts and attacks	1	France attack	2016-06-14–2016-08-14	173,025	3,318
	2	Orlando shooting	2016-05-12–2016-07-12	525,891	3,059
	3	Turkish coup	2016-06-15–2016-08-15	145,952	2,111
Disasters and accidents	4	California wildfire	2016-07-16–2016-09-16	44,954	407
	5	Hurricane matthew	2016-09-03–2016-11-03	570,124	1,604
	6	Louisiana flood	2016-07-17–2016-09-17	148,952	353
Business and economy	7	Federal reserve	2016-11-15–2017-01-15	54,213	2,141
	8	OPEC oil	2016-11-10–2017-01-10	116,429	1,450
	9	Trump TPP	2016-12-23–2017-02-23	60,798	470
Politics and elections	10	Trump protest	2016-12-20–2017-02-20	307,263	3,009
	11	Trump inauguration	2016-12-20–2017-02-20	525,149	6,082
	12	Trump tax	2017-03-26–2017-05-26	395,460	3,093
Arts and culture	13	Grammys	2017-01-12–2017-03-12	532,296	465
	14	Oscars	2017-01-26–2017-03-26	326,714	1,088
	15	Nobel prize	2016-09-08–2016-11-08	297,890	1,407
Sports	16	Super bowl	2017-01-06–2017-03-06	288,166	2,511
	17	Olympics	2016-07-13–2016-09-13	511,042	5,816
	18	NBA finals	2017-05-06–2017-07-06	254,798	936

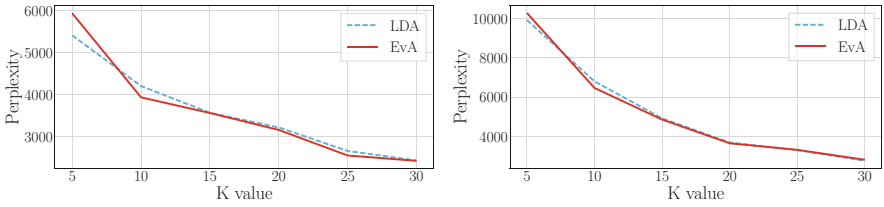
For data pre-processing, we remove stopwords and count the document (tweet) frequency for each word. We then remove words that appear in less than 10 documents for news and less than 20 tweets for Twitter. Finally, we obtain a vocabulary with 37,812 words for news media and 53,330 for Twitter.

4.2 Experiment

Experiment Configure. Regarding the parameters of our model, the prior for the hyper-parameters are set differently for news and Twitter. The values for δ, β, π are all set to 0.01 for news and 0.1 for Twitter, without further tuning, and λ_0^0, λ_1^0, λ_0^1, λ_1^1 are all set to 1, except that λ_1^0 set as 100 for news. The number of iterations for Gibbs sampling are fixed at 1000 for all methods. For baseline LDA, we set both α and β as 0.01 for news and 0.1 for Twitter.

Perplexity Comparison. We can rely on the perplexity as a measure to determine the most proper aspect number K. We thus present the results regarding the average word perplexity for different numbers of aspects. Given the estimated distribution q and the document set D_{test} for testing, we can compute the perplexity according to Eq. (5). A_d is the set of all aspect words in document d.

$$\text{perp}(D_{test} \mid q) = exp\{-\frac{\sum_{d \in D_{\text{test}}} \sum_{w \in A_d} logq(w \mid d)}{\sum_{d \in D_{\text{test}}} \mid A_d \mid}\}$$
$$q(w \mid d) = \sum_{k \in K} q(w \mid k) q(k \mid d) \tag{5}$$



(a) News (b) Twitter

Fig. 2. The perplexity results on news and Twitter with different K settings

The perplexity results are plotted in Fig. 2. The training and test splits are obtained by randomly selecting 80% documents as the training set, and reserving the rest for testing. From the results, we can observe that our model EvA outperforms the standard LDA. Considering that EvA also captures the category and background feature by separating words out from the aspect distribution, the corresponding number of the estimated $q(w \mid d)$ is lower than for LDA. Hence, the performance improvement is reasonable. In addition, we find that for $K = 10$, the perplexity results start to converge. We also observed that when the number of aspects for an event is larger than 10, there will be more repeated aspects. This is also consistent with the empirical characteristics of real-world events. Usually, for a single event, there are rarely more than 10 focus points. Therefore, we set $K = 10$ for the rest of the experiments.

4.3 Results Analysis

The event aspect words are responsible for capturing the content features, through which we can determine what points the mainstream news and Twitter users focus on. We obtain 10 event aspects for each event and select the top-10 words for each aspect. In Table 3, we list partial results of 3 aspects that best describe the differences in focal points between news and Twitter. We can derive the following conclusions:

1. Traditional news media tends to be more rational, impersonal, and serious than Twitter. Twitter users are often more emotional in expressing their personal feelings, e.g., praying in response to the Nice attack in France (A4 in Table 3), or for the Orlando shooting victims.

2. News media tends often considers the political background and implications. For instance, it reviews the previous Nobel Peace Prize winners (A3) and relates the gun control debate for the Orlando shooting with the elections.
3. The news media tend to be more comprehensive in fully covering relevant background and different aspects of an event. For instance, they provide relevant background knowledge to let the readers better understand the circumstances of an event, e.g., enumerating different countries relevant to "TPP" (A2). Twitter users care more about how the event may impact their personal life, e.g., users discuss the influence of the immigration ban policy on employment in US tech companies (A5).
4. News media shows a wider coverage of aspects, including important aspects that however may not directly affect a large percentage of their readership, e.g., the protest against the pipeline construction in Dakota (A1).
5. When discussing event-related topics, Twitter users focus more on major protagonists or celebrities. Twitter users tend to express their opinions about noteworthy people involved in the event (A6).

Table 3. Selected event aspect words in news and Twitter

News specific aspects			Twitter specific aspects		
A1	A2	A3	A4	A5	A6
Pipeline	Trade	ShimonPeres	Nice	Immigration	BobDylan
Dakota	USA	Prime	Terrorist	Ban	Literature
North	Countries	Minister	PrayForNice	Employees	Arrogant
Rock	Pacific	Israel	People	Google	Congrats
Standing	China	YitzhakRabin	Prayers	Tech	Impolite
Access	Deal	President	Victims	Christo	Winning
Army	Agreement	Peace	BastilleDay	Art	Called
Oil	Australia	Party	Sad	Comcast	Silence
Sioux	Pact	Leader	Mourning	World	Wind
Tribe	Zealand	YasserArafat	Families	Companies	Knock

Due to space constraints, we can't show the category words. However, we find that some words succeed in representing the important characteristics of a given category. For instance, *police*, *victims* in the attack category, *residents*, *damage* in the disaster category, *price*, *market* the in business category, etc.

5 Conclusion

In this paper, we have presented a new method EvA to analyze real-world events, combining news and social media. And we use this model to discover important

distinctions between mainstream news media and Twitter postings, based on the aspect analysis. In terms of future work, we will attempt to exploit our conclusions in several kinds of applications and follow-up studies. One direction is to consider classifications beyond the content-based category labels that we have considered thus far. Further kinds of classifications include those pertaining to the life cycle (short, middle, and long-term period), popularity (popular or not), authenticity (trustworthy or not), etc.

Acknowledgements. The authors wish to acknowledge the support provided by the National Natural Science Foundation of China (61503217, 91546203), the Key Research and Development Program of Shandong Province of China (2017CXGC0605) and China Scholarship Council (201606220187). Gerard de Melo's research is funded in part by ARO grant W911NF-17-C-0098 (DARPA SocialSim).

References

1. Abdelhaq, H., Sengstock, C., Gertz, M.: EvenTweet: online localized event detection from Twitter. Proc. VLDB Endow. **6**(12), 1326–1329 (2013)
2. Atefeh, F., Khreich, W.: A survey of techniques for event detection in Twitter. Comput. Intell. **31**(1), 132–164 (2015)
3. Castillo, C., El-Haddad, M., Pfeffer, J., Stempeck, M.: Characterizing the life cycle of online news stories using social media reactions. In: CSCW, pp. 211–223 (2014)
4. Hänska-Ahy, M., Wardle, C., Browne, M.: Social media & journalism: reporting the world through user generated content. Particip.: J. Audience Recept. Stud. **10**, 436–439 (2013)
5. Hoffart, J., Milchevski, D., Weikum, G.: STICS: searching with strings, things, and cats. In: SIGIR, pp. 1247–1248 (2014)
6. Hu, Y.: Event analytics on social media: challenges and solutions. Arizona State University (2014)
7. Hu, Y., Hong, Y.: Modeling Twitter engagement in real-world events. In: HICSS (2017)
8. Hu, Y., John, A., Seligmann, D.D., Wang, F.: What were the tweets about? Topical associations between public events and Twitter feeds. In: ICWSM (2012)
9. Hu, Y., John, A., Wang, F., Kambhampati, S.: ET-LDA: joint topic modeling for aligning events and their Twitter feedback. In: AAAI, vol. 12, pp. 59–65 (2012)
10. Li, C., Bendersky, M., Garg, V., Ravi, S.: Related event discovery. In: WSDM, pp. 355–364. ACM (2017)
11. Olteanu, A., Castillo, C., Diakopoulos, N., Aberer, K.: Comparing events coverage in online news and social media: the case of climate change. In: ICWSM, No. EPFL-CONF-211214 (2015)
12. Paul, M., Girju, R.: Cross-cultural analysis of blogs and forums with mixed-collection topic models. In: EMNLP, pp. 1408–1417. ACL (2009)
13. Setty, V., Anand, A., Mishra, A., Anand, A.: Modeling event importance for ranking daily news events. In: WSDM, pp. 231–240. ACM (2017)
14. Vieweg, S., Hughes, A.L., Starbird, K., Palen, L.: Microblogging during two natural hazards events: what Twitter may contribute to situational awareness. In: SIGCHI, pp. 1079–1088. ACM (2010)
15. Zhao, W.X., et al.: Comparing Twitter and traditional media using topic models. In: Clough, P., et al. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 338–349. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20161-5_34

Privacy

Differential Privacy for Regularised Linear Regression

Ashish Dandekar(✉), Debabrota Basu, and Stéphane Bressan

School of Computing, National University of Singapore, Singapore, Singapore
{ashishdandekar,debabrota.basu}@u.nus.edu, steph@nus.edu.sg

Abstract. We present ϵ-differentially private functional mechanisms for variants of regularised linear regression, LASSO, Ridge, and elastic net. We empirically and comparatively analyse their effectiveness. We quantify the error incurred by these ϵ-differentially private functional mechanisms with respect to the non-private linear regression. We show that the functional mechanism is more effective than the state-of-art differentially private mechanism using input perturbation for the three main regularised linear regression models. We also discuss caveats in the functional mechanism, such as non-convexity of the noisy loss function, which causes instability in the results.

Keywords: Linear regression · Data privacy · Differential privacy

1 Introduction

Dwork et al. proposed *differential privacy* [5] to quantify the privacy of a mechanism.

Let $\mathcal{D}$ denotes a universe of d-dimensional real-valued datapoints and the corresponding real-valued responses of a statistical machine learning algorithm M. An element from this universe can be represented by a pair $D = (X, Y)$ where $X \in \mathbb{R}^{n \times d}$ is a matrix and $Y \in \mathbb{R}^n$ is a vector. We use $\|\cdot\|_p$ to represent the L_p norm of a vector. Let us call a *neighbouring* dataset a dataset D' that differs from the dataset D by one datapoint. Differential privacy quantifies the privacy of a randomized algorithm, referred to as a mechanism $\mathcal{M}$, run on those datasets.

Definition 1. *[5] A randomized algorithm $\mathcal{M}$ with domain $\mathcal{D}$ is ϵ-differentially private if for all $S \in Range(\mathcal{M})$ and $D, D' \in \mathcal{D}$ such that D and D': are neighbouring datasets*

$$\log\left(\left|\frac{Pr(\mathcal{M}(D) \in S)}{Pr(\mathcal{M}(D') \in S)}\right|\right) \leq \epsilon$$

Differential privacy introduces noise at selected stages of a statistical machine learning algorithm, for instance in the output of the model [4] or in the input of the algorithm [10] or in the loss function as with the *functional mechanism* [17]. The calibration of these mechanisms requires the computation of *sensitivity* of the function.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 483–491, 2018.
https://doi.org/10.1007/978-3-319-98812-2_44

Definition 2. *The sensitivity of a function* $f : \mathcal{D} \rightarrow \mathbb{R}^k$ *is defined as:*

$$\Delta_f = \max_{x \sim y} \|f(x) - f(y)\|_1$$

Laplace mechanism [4] is a widely used privacy-preserving mechanism. It achieves differential privacy by adding random noise from a Laplace distribution. For a given privacy level ϵ, Laplace distribution is calibrated such that the mean is zero and the scale $\frac{\Delta_f}{\epsilon}$. The Laplace mechanism calibrated in this way satisfies ϵ-differential privacy [5].

We present ϵ-differentially private functional mechanisms for variants of regularised linear regression, namely LASSO, Ridge, and elastic net. Linear regression [11] is a widely used statistical machine learning model. It uses a linear hypothesis to map a set of predictor attributes of a datapoint to the corresponding response. In matrix notation, linear regression is parameterized by $\theta \in \mathbb{R}^d$ such that $X\theta = Y$. In order to find the optimal value of θ, training step in linear regression minimizes *mean squared loss*, $l_\theta(T)$, over the training data T, as defined in Eq. 1.

$$\theta^* = \arg\min_\theta l_\theta(T) = \arg\min_\theta (X\theta - Y)^2 \tag{1}$$

Properties of the coefficient of quadratic term in Eq. 1 determine the convexity of the optimisation problem. The optimization problem is made convex by adding a regularisation term to the objective function in Eq. 1. With a regularisation term that is proportional to the L_2 norm of the parameters the new optimisation is called *Ridge regression* [9]. It is defined in Eq. 2.

$$\theta^* = \arg\min_\theta (X\theta - Y)^2 + \lambda \|\theta\|_2^2 \tag{2}$$

With a regularisation term that is proportional to the L_1 norm of the parameters the new optimisation is called *LASSO regression* [15]. With a regularisation term that is proportional to a convex combination of L_1 and L_2 norms of parameters the new optimisation is called *Elastic net regression* [18].

In Sect. 3, we extend the work of the authors of [17] for Ridge regression and present the functional mechanism [17] for linear regression and three of its regularised variants, namely, Ridge, LASSO and Elastic net.

In Sect. 4, we comparatively evaluate the performance of these four mechanisms and their differentially private variants on two datasets with different correlations and sparsity. We observe that the functional mechanism applied to the regularised linear regression yields similar performance results and that the private linear regression models perform worse than the non-private linear regression models. We compare the effectiveness of the functional mechanism with an *input perturbation mechanism* [10]. For a given privacy level, ϵ, we empirically show, for the three main regularised linear regression models, that the functional mechanism is more effective than the state-of-art differentially private mechanism using input perturbation, DPME [10]. We extend the analysis in [17] to empirically study the robustness of the functional mechanism. The key

observation in our experiments is that all the private linear regression models are unstable. Our analysis shows that the reason for such an instability is inherent to the functional mechanism. In reference to these experimental evidences, we conclude by puting forth (Sect. 5) the need of designing a differentially private mechanism that produces a convex noisy loss function in order to provide both stable and private output for linear regression models.

The extended version of this paper is available at [2].

2 Related Work

Linear regression [11] is a fundamental yet a widely used machine learning model. Variants of linear regression, Ridge [9] and LASSO [15], are used to reduce correlation in the data features and to avoid overfitting. Elastic net [18] regression uses convex combination of regularisation terms that are used in Ridge and LASSO. For a detailed presentation and discussion of regularisation and regression analysis, interested readers can refer to [11].

Differential Privacy [5] is a probabilistic framework that quantifies the privacy of a randomized function or algorithm. Existing deterministic machine learning models can be randomized by introducing calibrated random noise. The resultant randomized *mechanism* can be shown to satisfy constraints of differential privacy. Dwork et al. propose the Laplace mechanism [4], which perturbs the output of a machine learning model by explicitly adding scaled random noise from the Laplace distribution. The Gaussian mechanism [5] and the K-norm mechanism [8] are differentially private mechanisms that are also based on the idea of output perturbation with noise from different distributions. Lei [10] proposes differentially private M-estimators, which perturbs the histogram of input data using a scaled noise and further uses the noisy histogram to train the models. Zhang et al. [17] propose a differentially private *functional mechanism* that adds a properly scaled Laplace noise to the coefficients of loss function in the polynomial basis. Hall et al. [6] also propose a differentially private *functional mechanism* that adds a properly scaled noise drawn from the Gaussian process to the coefficients of loss function in the kernel basis.

Zhang et al. instantiate their functional mechanism on linear regression and logistic regression. In order to alleviate the non-convexity caused in the loss function due to addition of random noise, they use Ridge regularised linear and logistic regressions. Yu et al. [16] achieve differential privacy in the elastic net logistic regression by controlling the coefficient of regularisation term. The regularisation term in their proposal is inversely proportional to the number of datapoints. It causes reduction in regularisation as the number of datapoints increases. Therefore, their proposed mechanism is not applicable for large datasets. Talwar et al. [13] propose a differentially private variant of Frank-Wolfie optimisation algorithm to perform LASSO regression. This method adds noise in the optimisation algorithm instead of adding it to the objective function.

3 Functional Mechanism for Regularised Linear Regression

Functional mechanism [17], which is a privacy-preserving mechanism, introduces random noise in the loss function of a machine learning algorithm. Optimisation of such a noisy loss function leads to the parameters that are different than true optimal parameters. In this way, we indirectly get noisy outputs from the machine learning model without explicitly adding noise to the outputs. In this section, we elucidate the details related to the functional mechanism.

For a given machine learning model, the loss function l_θ can be expanded in the polynomial basis, using Stone-Weierstrass theorem [14], as a function of parameter θ as given in Eq. 3 where $t = (x, y)$ denotes a datapoint in training dataset T, Φ_j denotes the set of polynomials with degree j and $\lambda_{t\phi}$ denote respective coefficients.

$$l_\theta(T) = \sum_{t \in T} \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \lambda_{t\phi} \phi(\theta) \tag{3}$$

Lemma 1. *[17] Upper bound on sensitivity of the loss function of a machine learning model is given by:*

$$\Delta_l = 2 \max_t \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \|\lambda_{t\phi}\|_1$$

Using Lemma 1 and the Laplace mechanism, Zhang et al. [17] devise an algorithm, namely the functional mechanism, that adds noise to loss function of a machine learning model. For a given privacy level ϵ, they use Laplace mechanism calibrated with the sensitivity calculation in Lemma 1 to induce noise in the coefficients of the Taylor expansion of the loss function. Parameters of the machine learning model are estimated by optimizing the noisy loss function. They prove that this algorithm satisfies ϵ-differential privacy. Please refer to Algorithm 1 in [17] for the details.

Elastic net regression [18] adds the regularisation term which is a convex combination of L_1 regularisation term and L_2 regularisation term. The optimisation problem for elastic net regression with functional mechanism is given in Eq. 4. $l'_\theta(T)$ denotes the noisy loss function obtained by applying the functional mechanism on the loss function for linear regression, as stated in Eq. 1.

$$\theta^* = \arg\min_\theta l'_\theta(T) + \lambda(\alpha\|\theta\|_2^2 + (1-\alpha)\|\theta\|_1) \tag{4}$$

The additive regularisation term is proportional to the norm of the parameters and it does not depend on the training dataset. Therefore, regularisation does not change the sensitivity of the loss function. We use this observation and the sensitivity calculation for linear regression in [17] to compute the sensitivity of Elastic net regression. We present the result below.

Assuming that all features of datapoints are normalized such that each of the feature value lies in $[-1, 1]$, L_1 sensitivity of the loss function in Eq. 4 is given by:

$$\Delta_l = 2(d^2 + 2d + 1)$$

4 Empirical Performance Evaluation

We comparatively and empirically evaluate functional mechanism for regularised linear regressions: namely Ridge, LASSO, and elastic net. We present the result analysis in this section.

We conduct experiments on a microdata sample of US Census in 2000 provided by IPUMS International [1]. The census dataset consists of 1% sample of the original census data. We consider a subset of 316,276 records of the heads of households in our dataset. Each record has 9 attributes, namely, *Age, Gender, Race, Marital Status, Family Size, Education, Employment Status, House type, Income*. Regression analysis is performed using *Income* as the response variable and the rest of the attributes as predictor variables.

We use Python® 2.7.6 with the SCS [12] solver from CVXPY [3] package.

We report the results as the aggregates over 50 experimental runs. For every experimental run, we randomly hold out 20% of the data for testing and use the rest 80% of the data for training regression models. We normalize each of these features such that their values lie in $[-1, 1]$. We use *root mean squared error (RMSE)* [11] as the metric to comparatively evaluate effectiveness. For given value of ϵ, the model with smallest value of RMSE is the most effective model.

We comparatively evaluate eight regression problems: linear regression (LR), Ridge regression (RG), LASSO regression (LS), elastic net regression (EN), and their private versions. We call the regression model obtained using the functional mechanism *functional regression*. For every regularised regression model, we set, by cross-validation, the regularisation coefficient, λ, that yields smallest testing error.

Figure 1 shows the comparative evaluation of the functional mechanisms with an *input perturbation mechanism*, differentially private M-estimators (DPME) [10]. Discretisation of a large number of attributes leads to a large discrete space that causes prohibitive computation cost. Due to concentration of data around subsets of features, a large discrete space also leads to sparse histograms [10]. In order to alleviate the sparsity, we follow [10] and evaluate the performance on a simpler regression model. We show the comparative study on the census dataset where we predict *Income* of a person using *Age, Gender, Race* and *Education Status*. The results are presented in Fig. 1. Solid lines represent the *mean* RMSE over 50 runs. For a given value of ϵ, we observe that the functional mechanism provides lower RMSE for all regularised linear regressions. Thus, we show that the functional mechanism is more effective than DPME.

Now we present the comparative evaluation functional regularised regressions. Figure 2 shows the boxplot of functional elastic net regression for different values of ϵ's. We note the presence of a large number of outliers in the result.

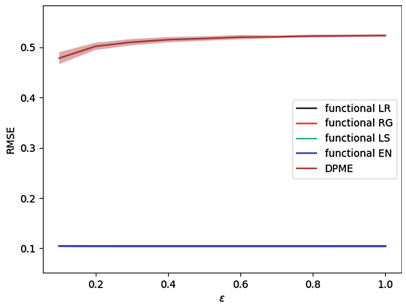


Fig. 1. RMSE of regularised linear regressions for varying values of ϵ's for DPME [10] and the functional mechanism

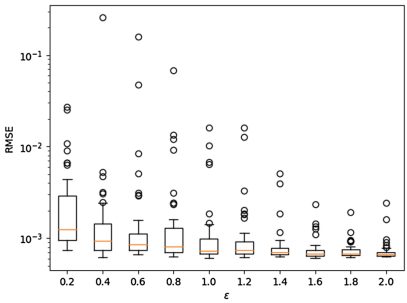


Fig. 2. Boxplot of RMSE of elastic net Ridge regression with functional mechanism for different values of ϵ for the census dataset.

We observe similar results for the rest of the functional regressions. In order to avoid this bias due to the outliers, we choose to plot the *median* instead of the *mean*. Figure 3 shows the comparative evaluation of the variants of regularised linear regression for the census quality dataset. In the plot, solid line represents median over 50 experimental runs and the shaded region covers RMSE values

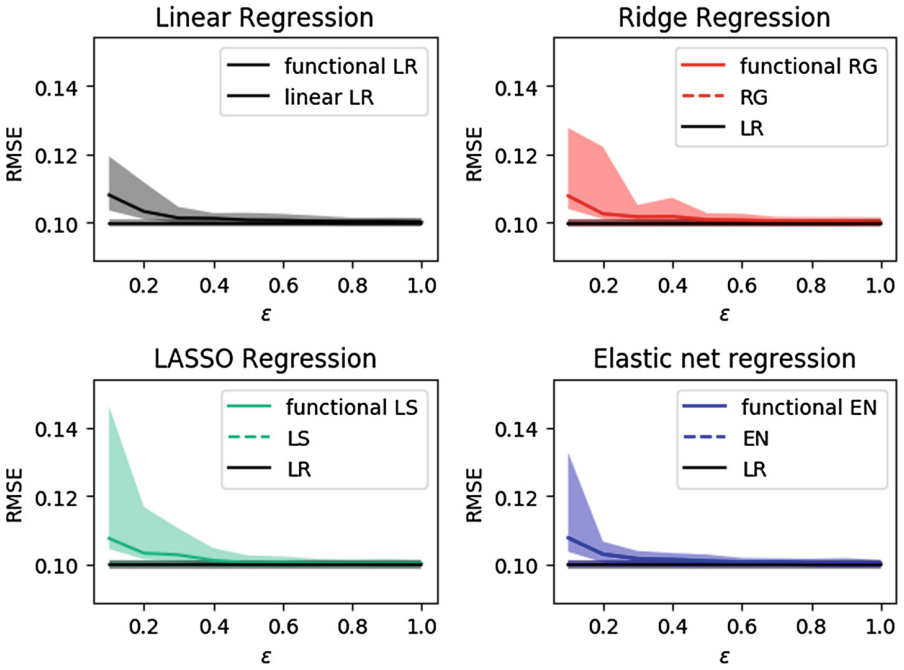


Fig. 3. Comparison of different regressions for the census dataset with *median* as the aggregate

that lie between 20^{th} and 80^{th} percentile. Smaller values of ϵ's induce higher noise in the function, which in turn results in higher privacy. Therefore, we observe higher RMSE for smaller values of ϵ's. As the value of ϵ increases, the effectiveness of the functional regression approaches the the effectiveness of the non-private counterpart.

One observation that is common in all the empirical evaluation in Fig. 2 and Fig. 3 is the instability in the results. M-estimator is a robust statistic [7]. We observe the stability in the performance of DPME as compared to the functional regressions. We find that the reasons for this instability are rooted in the functional mechanism itself. We complete the result analysis by the discussion of these possible reasons.

The coefficient of the quadratic term in Eq. 1, X^tX, is a symmetric matrix. It loses its symmetric property after adding random noise from the Laplace distribution. A standard way to make a given matrix A symmetric is to use $(A + A^t)$ * 0.5. This way of symmetrization of noisy X^tX indirectly incurs addition two Laplace random variables. Addition of two Laplace random variable does not follow Laplace distribution. Therefore, in order to maintain the integrity of the functional mechanism, we can not make X^tX symmetric in the conventional way.

Linear regression works on the assumption that the attributes in a dataset are independent of each other. Independence among the attributes makes X^tX a positive definite matrix. Positive definite matrices make the optimisation convex and guarantees optimality of the solution. Noisy loss function fails to guarantee convexity of the objective problem, and hence the optimality of the solution. A similar observation is made by Lei [10] while perturbing the histograms of input data by adding the calibrated noise. In order to make the objective function convex, Zhang et al. [17] calculate the spectral decomposition of X^tX and consider the projection of parameters onto the eigenspace spanned by eigenvectors with positive eigenvalues. They do not provide any analytical justification which guarantees differential privacy after pruning the non-positive eigenspace.

Functional mechanism proves that the loss function generated by any two neighbouring datasets satisfies differential privacy. Composition of a differentially private function with a deterministic function, called as post-processing [5], remains differentially private. An optimisation problem solver calculates an approximate solution when the objective function is not convex. Therefore, differential privacy of a loss function is not preserved by the optimisation algorithm itself.

5 Conclusion and Future Works

We present the construction of differentially private versions of the three linear regression models, Ridge, LASSO and Elastic net, using the functional mechanism. We empirically and comparatively evaluate the effectiveness of the private and non-private versions on a census datasets. For a given privacy level, ϵ, we observe that the functional mechanism is more effective than DPME [10] for

regularized linear regression. We extend the analysis in [17] to empirically study the robustness of the functional mechanism. As expected, we invariably observe that the private versions are less effective than their non-private counterparts. The key observation from these experiments is that all these private regularised regression methods are equally unstable, and that private linear regression is comparatively more unstable. We analyse the loss of symmetry of the covariance matrix and the non-convexity of the loss function after adding the noise as the principal reasons of this instability. This opens up the need of designing a privacy-preserving mechanism that would retain these properties for private linear regression.

Acknowledgement. This research is supported by the National Research Foundation, Prime Minister's Office, Singapore, under its Corporate Laboratory@University Scheme, National University of Singapore, and Singapore Telecommunications Ltd.

References

1. Minnesota Population Center. Integrated public use microdata series - international: version 5.0 (2009). https://international.ipums.org
2. Dandekar, A., Basu, D., Bressan, S.: Differential privacy for regularised linear regression. Technical report TRA6/18, National University of Singapore, June 2018. https://dl.comp.nus.edu.sg/handle/1900.100/7051
3. Diamond, S., Boyd, S.: CVXPY: a Python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(83), 1–5 (2016)
4. 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
5. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Found. Trends® Theoret. Comput. Sci. **9**(3–4), 211–407 (2014)
6. Hall, R., Rinaldo, A., Wasserman, L.: Differential privacy for functions and functional data. J. Mach. Learn. Res. (JMLR) **14**(Feb), 703–727 (2013)
7. Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A.: Robust Statistics: The Approach Based on Influence Functions, vol. 196. Wiley, London (2011)
8. Hardt, M., Talwar, K.: On the geometry of differential privacy. In: Proceedings of the Forty-Second ACM Symposium on Theory of Computing (STOC). ACM (2010)
9. Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)
10. Lei, J.: Differentially private M-estimators. In: Advances in Neural Information Processing Systems, pp. 361–369 (2011)
11. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
12. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**, 1042–1068 (2016)
13. Talwar, K., Thakurta, A.G., Zhang, L.: Nearly optimal private LASSO. In: Advances in Neural Information Processing Systems (NIPS), pp. 3025–3033 (2015)
14. Thomas, G.B., Weir, M.D., Hass, J.: Thomas Calculus. Addison-Wesley, Reading (2016)

15. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. R. Stat. Soci. Ser. B (Methodol.) **58**, 267–288 (1996)
16. Yu, F., Rybar, M., Uhler, C., Fienberg, S.E.: Differentially-private logistic regression for detecting multiple-SNP association in GWAS databases. In: Domingo-Ferrer, J. (ed.) PSD 2014. LNCS, vol. 8744, pp. 170–184. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11257-2_14
17. Zhang, J., Zhang, Z., Xiao, X., Yang, Y., Winslett, M.: Functional mechanism: regression analysis under differential privacy. Proceed. VLDB Endow. **5**(11), 1364–1375 (2012)
18. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Stat. Soci.: Ser. B (Stat. Methodol.) **67**(2), 301–320 (2005)

A Metaheuristic Algorithm for Hiding Sensitive Itemsets

Jerry Chun-Wei Lin[1,2(✉)], Yuyu Zhang[1], Philippe Fournier-Viger[3], Youcef Djenouri[4], and Ji Zhang[5]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen, China
yuyuzhang.hit@gmail.com, jerrylin@ieee.org
[2] Department of Computing, Mathematics, and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
[3] School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen, Shenzhen, China
philfv8@yahoo.com
[4] IMADA, Southern Denmark University, Odense, Denmark
djenouri@imada.sdu.dk
[5] University of Southern Queensland, Toowoomba, QLD, Australia
Ji.Zhang@usq.edu.au

Abstract. In this paper, we first present a multi-objective algorithm for hiding the sensitive information with transaction deletion based on the NSGAII framework. The proposed can efficiently sort the non-dominated solutions and find the set of Pareto results for later process. Experimental results on two real datasets illustrated that the proposed algorithm can achieve satisfactory results with fewer side effects compared to the previous single-objective evolutionary approaches.

Keywords: PPDM · Sanitization · Evolutionary computation
Pareto solutions

1 Introduction

In recent years, Knowledge Discovery in Database (KDD) [8–10] is important approach for mining the potential and implicit relationships from databases. Although data mining techniques can be used to reveal the implicit information from a very large database, the private or secure data may, however, also be easily explored and discovered. Privacy-preserving data mining (PPDM) has thus become a critical issue in recent years [2,15], which is used to aggregate the data for analytics while reducing the privacy threats by hiding sensitive information. Several algorithms were respectively studied [3,12,18] to preserve the confidential information but still discover to find the useful information.

The traditional algorithms of PPDM are difficult to find the optimal transactions/items to be sanitized for minimizing the side effects. Thus, several algorithms [4,16,17] were proposed to optimize the sanitization procedure based on

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 492–498, 2018.
https://doi.org/10.1007/978-3-319-98812-2_45

evolutionary algorithms (GAs) [11], which can facilitate for finding the optimal solutions based on the principles of natural evolution. In order to find better transactions to be deleted for hiding sensitive information, a NSGAII-based algorithm with transaction deletion is thus presented. Based on the designed approach, the non-dominated chromosomes can be easily discovered, and the better solutions can be obtained for later transaction deletion. Experiments also showed that the designed algorithm can achieve better performance in terms of four side effects in most cases.

2 Literature Review

In early 1970s, Holland applied the Darwin theory of natural selection and survival of the fittest, into the field of dynamic algorithms and proposed the evolutionary computation of genetic algorithms (GAs) [11]. Variants of single-objective evolutionary computation were respectively studied, for example, Particle Swarm Optimization (PSO) [13], Ant Colony Optimization (ACO) [19], Differential Evolutionary [21], etc. In real-world situations, optimization should consider the multiple objectives for finding the set of solutions. The Non-dominated Sorting Genetic Algorithm (NSGA) [23] considers the multiple objectives and functions for optimization, which is an extension of traditional GAs. The NSGAII [5] was proposed by Deb et al., which is inspired by elder NSGA algorithm. Several variants of evolutionary multiple objective optimization were respectively presented [14,22,25].

Privacy-preserving data mining (PPDM) [2] has become an important issue in recent years since it can be used to hide the sensitive information but still keeps the data integrity as much as possible to find the useful and meaningful information. Verykios et al. presented a classification hierarchy of PPDM techniques [24]. Dasseni et al. then proposed a hiding approach based on the hamming-distance method, which is used to decrease the confidence or support values of association rules for hiding sensitive information [6]. Several algorithms of PPDM are still developed in progress [1,20].

Since the sanitized algorithms of PPDM can be concerned as the NP-hard problem [24], meta-heuristic algorithms can obtain better and suitable solutions to solve this limitation. Lin et al. respectively presented the sGA2DT, pGA2DT [17] and cpGA2DT [16] algorithms for hiding the sensitive itemsets by removing the victim transactions based on GAs. Peng Cheng et al. [4] proposed the EMO-RH algorithm for hiding the sensitive itemsets by removing the itemset based on EMO. However, this approach causes seriously problem with the incomplete information discovered for decision making.

3 Preliminaries and Problem Statement

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items occurring in the database D, and D is a set of transactions where $D = \{T_1, T_2, \ldots, T_n\}$, $T_q \in D$. Each T_q is a subset of I and is defined as *TID*. The set of sensitive itemsets

is denoted as $SI = \{s_1, s_2, \ldots, s_k\}$, which can be defined by users' preference. Note that each $s_l \in SI$ is also a subset of T_q. If $sup(i_j) \geq S_u$, the itemset (i_j) is defined as the frequent (large) itemset (in the set of L). The S_u is defined as the traditional minimum support threshold.

Definition 1. *For each $s_l \in SI$, the number of transactions for deletion of s_l is denoted as DL_{s_l} and defined as:*

$$DL_{s_l} > \frac{sup(s_l) - S_u \times |D|}{1 - S_u}. \tag{1}$$

Definition 2. *The maximum number of deleted transactions among all sensitive itemsets in SI is denoted as DL_{max} and defined as:*

$$DL_{max} = max\{DL_{s_1}, \ldots, DL_{s_k}\}. \tag{2}$$

The DL_{max} is treated as the size of the chromosome in the designed approach. In the PPDM, the side effects are respectively defined as "hiding failure", "missing cost", and "artificial cost". We can obtain the definitions for three side effects as follows.

Definition 3. *Let α be the number of sensitive itemsets that fail to be hidden, and L' be the number of large itemsets after the sanitization process, we can have that:*

$$\alpha = |SI \cup L'|. \tag{3}$$

Definition 4. *Let β be the number of missing itemsets (cost), which is the non-sensitive itemset and the large itemset in the original database but will be hidden after the sanitization progress as:*

$$\beta = L - SI - L'. \tag{4}$$

Definition 5. *Let γ be the number of artificial itemsets (cost), which was not the frequent itemset but will be arisen as the frequent one after the sanitization progress as:*

$$\gamma = L' - L. \tag{5}$$

Besides the traditional three side effects, the similarity between the original database and the sanitized one should also be considered as one of major criteria to evaluate the performance of the sanitization algorithm in PPDM [17].

Definition 6. *Let Dis be the dissimilarity between the original database and the sanitized one, which can be defined as:*

$$Dis = |D| - |D^*|, \tag{6}$$

where $|D|$ is the size of the original database, and $|D^*|$ is the sanitized database.

From the perspective of optimization in PPDM, it is better to find the suitable transactions for deletion to hide the sensitive information and achieve the minimal side effects. Thus, we obtain the NSGAII framework with transaction deletion to handle the problem of PPDM by considering multiple objective functions.

Problem Definition: The problem of PPDM with transaction deletion based on the NSGAII framework is to minimize four side effects but still hide the sensitive information as much as possible, which can be defined as:

$$f = min[f_1, f_2, f_3, f_4], \tag{7}$$

where $f_1 = \alpha$, $f_2 = \beta$, $f_3 = \gamma$, and $f_4 = Dis$.

4 Proposed Algorithm for Sanitization

To hide the sensitive itemset through transaction deletion, the support of each sensitive itemset should below the upper support threshold. Thus, only the transactions consist of sensitive information should be processed and determined. In the designed algorithm, we first project the transactions containing any of sensitive itemset, and denote the projected database as D^*. A chromosome consists of the number of DL_{max} genes as a solution. Each gene is equal to the transaction ID from D^*. Also, each gene can be a ***null*** value in the designed chromosome. The designed algorithm is given as below.

Designed Algorithm:

STEP 1: Find the chromosome size of DL_{max}.
STEP 2: Scan database D to obtain the large itemsets L.
STEP 3: Project the transactions with any of sensitive itemsets si as D^*.
STEP 4: Initial the chromosomes randomly, and each gene is from D^*.
STEP 5: Perform the *mutation*, *crossover*, and *selection* of the chromosomes.
STEP 6: Evaluate each chromosome by four fitness functions using Eq. (7).
STEP 7: Find the set of Pareto solutions.
STEP 8: Sort the derived solutions by the *Pareto-rank* mechanism.
STEP 9: Select the solutions as follows:
- **STEP 9-1:** Select the chromosomes as the solutions from first property to last.
- **STEP 9-2:** If the chromosomes are in the same rank, use the crowding-distance to obtain it.
- **STEP 9-3:** Find the top-N chromosomes for next generation.

STEP 10: Perform the steps 4 to 10 until the termination criteria ia achieved.

5 Experimental Results

Substantial experiments were conducted to verify the effectiveness and efficiency of the proposed algorithm compared to the state-of-the-art cpGA2DT [16] and PSO2DT [18] approaches under chess and foodmart datasets [7]. The weightings for three side effects α, β, and γ are respectively set as 0.8, 0.1, and 0.1 for the cpGA2DT and PSO2DT algorithms. The results under varied percentages of *SI* are shown in Table 1.

Table 1. Results under different percentages of SI.

Dataset	% of SI	Evaluation (%)	Proposed algorithm	cpGA2DT	PSO2DT
Chess	4	α	26.4	25.0	**0**
		β	**3.24**	9.23	17.1
		γ	**0.32**	0.48	0.646
		Dis	**8.45**	14.6	14.7
	6	α	36.25	33.33	**0**
		β	**4.71**	10.06	21.4
		γ	0.96	0.16	**0**
		Dis	**7.45**	12.2	12.2
	8	α	43.23	40.0	**0**
		β	**2.12**	9.56	23.5
		γ	0.96	2.79	**0**
		Dis	**7.82**	14.4	14.2
Foodmart	6	α	8.92	10.0	**0**
		β	**0**	1.25	1.25
		γ	**0**	**0**	**0**
		Dis	**0.26**	0.45	0.34
	10	α	**10.0**	**10.0**	**10.0**
		β	1.56	2.5	**0**
		γ	**0**	**0**	**0**
		Dis	**0.29**	0.41	0.35
	14	α	15.79	14.29	**14.29**
		β	**1.3**	3.1	2.8
		γ	**0**	**0**	**0**
		Dis	**0.35**	0.41	0.41

In Table 1, it is obvious to see that the proposed algorithm mostly has lower β, γ, and even the *Dis* on the chess and the foodmart datasets. For most cases under varied sensitive percentages, the proposed algorithm still has good performance in terms of missing cost ($=\beta$) and data dissimilarity ($=Dis$) in chess

dataset. We also can obtain that the designed algorithm outperforms the other two algorithms in terms of β, γ, and *Dis* in the foodmart dataset. The reason is that in the conducted experiments, the weight of hiding failure is set very high, for example, 0.9%; thus the hiding failure of those two algorithms is less than the designed algorithm since it is the first property to be considered in their PPDM procedures. However, the designed algorithm focuses on providing a set of solutions by considering four objectives without any pre-defined fitness function, it is more flexible and realistic than the other two single-objective algorithms.

6 Conclusion

In this paper, we introduce an algorithm for hiding the sensitive information based on the NSGAII framework with transaction deletion. The designed algorithm can achieve better results by considering four major side effects in the PPDM. Experiments showed that the designed algorithm outperforms the single-objective evolutionary algorithms.

Acknowledgment. This research was partially supported by the Shenzhen Technical Project under JCYJ20170307151733005 and KQJSCX20170726103424709.

References

1. Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., Verykios, V.: Disclosure limitation of sensitive rules. Workshop Knowl. Data Eng. Exch. **29**, 45–52 (1999)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. ACM SIGMOD Rec. **29**(2), 439–450 (2000)
3. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explor. **4**, 1–7 (2003)
4. Chen, P., Lee, I., Lin, C.W., Pan, J.S.: Association rule hiding based on evolutionary multi-objective optimization. Intell. Data Anal. **20**(3), 495–514 (2016)
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., et al. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45356-3_83
6. Dasseni, E., Verykios, V.S., Elmagarmid, A.K., Bertino, E.: Hiding association rules by using confidence and support. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137, pp. 369–383. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45496-9_27
7. Fournier-Viger, P., et al.: The SPMF open-source data mining library version 2. In: Berendt, B., et al. (eds.) ECML PKDD 2016. LNCS (LNAI), vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8
8. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. Data Sci. Pattern Recogn. **1**(1), 54–77 (2017)
9. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H. C., Tseng, V.S., Yu, P.S.: A survey of utility-oriented pattern mining. arXiv:1805.10511 (2018)
10. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H. C., Yu, P. S.: A survey of parallel sequential pattern mining. arXiv:1805.10515 (2018)

11. Holland, J.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. MIT Press, Cambridge (1992)
12. Han, S., Ng, W.K.: Privacy-preserving genetic algorithms for rule discovery. In: Song, I.Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 407–417. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74553-2_38
13. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
14. Knowles, J., Corne, D.: The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: The Congress on Evolutionary Computation, pp. 98–105 (1999)
15. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: The Annual International Cryptology Conference on Advances in Cryptology, pp. 36–54 (2000)
16. Lin, C.W., Zhang, B., Yang, K.T., Hong, T.P.: Efficiently hiding sensitive itemsets with transaction deletion based on genetic algorithms. Sci. World J. **2014**, 13 (2014). Article ID 398269
17. Lin, C.W., Hong, T.P., Yang, K.T., Wang, S.L.: The GA-based algorithms for optimizing hiding sensitive itemsets through transaction deletion. Appl. Intell. **42**(2), 210–230 (2015)
18. Lin, J.C.W., Liu, Q., Fournier-Viger, P.: A sanitization approach for hiding sensitive itemsets based on particle swarm optimization. Eng. Appl. Artif. Intell. **53**, 1–18 (2016)
19. Marco, D., Sabrina, O., Thomas, S.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2004)
20. Oliveira, S.R.M., Zaïane, O.R.: Privacy preserving frequent itemset mining. In: IEEE International Conference on Privacy, Security and Data Mining, pp. 43–54 (2002)
21. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
22. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: International Conference on Genetic Algorithms, vol. 2, no. 1, pp. 93–100 (1985)
23. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. **2**(3), 221–248 (1994)
24. Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y.: State-of-the-art in privacy preserving data mining. ACM SIGMOD Rec. **33**, 50–57 (2004)
25. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)

Text Processing

Constructing Multiple Domain Taxonomy for Text Processing Tasks

Yihong Zhang[1(✉)], Yongrui Qin[2], and Longkun Guo[3]

[1] Graduate School of Informatics, Kyoto University, Kyoto, Japan
yihong.zhang.8z@kyoto-u.ac.jp
[2] School of Computing and Engineering, University of Huddersfield, Huddersfield, UK
y.qin2@hud.ac.uk
[3] College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
lkguo@fzu.edu.cn

Abstract. In recent years large volumes of short text data can be easily collected from platforms such as microblogs and product review sites. Very often the obtained short text data contains several domains, which poses many challenges in effective multi-domain text processing because it is challenging to distinguish among the multiple domains in the text data. The concept of *multiple domain taxonomy* (MDT) has shown promising performance in processing multi-domain text data. However, MDT has to be constructed manually, which requires much expert knowledge about the relevant domains and is time consuming. To address such issues, in this paper, we introduce a semi-automatic method to construct an MDT that only requires a small amount of manual input, in combination of an unsupervised method for ranking multi-domain concepts based on semantic relationships learned from unlabeled data. We show that the iteratively-constructed MDT using our semi-automatic method can achieve higher accuracy than existing methods in domain classification, where the accuracy can be improved by up to 11%.

1 Introduction

In recent years, microblogs such as Twitter and Tumblr that use short text as the main media have gained huge popularity, with millions of short text messages exchanged every day. While the general purpose for a microblog is to allow communication between users and followers, the short texts in microblogs have been mined for various specific purposes, such as detecting earthquakes, monitoring influenza, and predicting election results [3,10,13].

Messages collected from Twitter, called tweets, are usually a mixture of conversations, announcements, and comments in various topic domains. To use tweets for a specific purpose, one cannot avoid the step to filter the tweets for a target domain, before the data can be used [6]. A problem with current text processing approaches is that the solution for one domain generally cannot be used in another domain [1]. And those works that aim at providing a multi-domain solution are usually limited to well-known domains such as news, sports,

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 501–509, 2018.
https://doi.org/10.1007/978-3-319-98812-2_46

entertainment, etc. [8]. Therefore it is novel and challenging to provide a general solution that can be easily adopted for less-known specific domains.

In our previous work, we show that *multiple domain taxonomy* (MDT) is effective for solving text processing problems [15]. An MDT is a special ontology that has two types of relationships, namely, *domain association* and *taxonomy association*, with each concept in MDT assigned to a domain and a taxonomy. Our previous work builds MDTs manually, while in this work, we present a semi-automatic construction method that allows an MDT to be built with relative ease. To summarize, our contributions with this paper include:

- We present a semi-automatic method for constructing MDTs. From some initial concepts, this construction method iteratively finds discriminative terms from unlabeled data, and ranks them to suggest potential concepts. It only requires a small amount of manual effort in providing initial concepts, and checking top-k terms in each construction iteration.
- We conduct experiments for testing the effectiveness of using iteratively constructed MDTs for text domain classification. With real-world tweet datasets, the experimental results show that our method can achieve high accuracy in text domain classification tasks, and can outperform existing methods by up to 11%.

2 Related Work

Given the emerging popularity of social media, text processing tasks such as domain classification has been widely studied. Sriram et al. [11] propose a classification method to identify message categories such as news, opinions, and deals. Targeting such categories, their method is a supervised learning approach based on text features such as opinion words, time-event phrases, and the use of dollar sign. Li et al. [6] propose a classification method to find the Crime and Disaster Events (CDE). They use a supervised classifier that incorporates features that include hashtag, URL, and CDE-specific features such as time mention. They found that including CDE features provides about 80% accuracy versus 60% accuracy without them. Sakaki et al. [10] also use a supervised classifier to filter earthquake observations, and includes features such as the number of words in the tweet and the index of the keyword. A common problem of these works is that a solution that works well in one domain may not work in other domains.

A few existing works focus on constructing ontology from short text and tweets, and then use the ontology for matching and classification purposes [5,8,12]. Lucia et al. [8] propose an unsupervised message classification method based on expanding lexical meanings using external knowledge sources. Their work, however, is restricted to general categories such as business, politics and arts, due to the knowledge base they use. Suliman et al. [12] propose an ontology-based event assertion method. They first choose initial keywords using latent topic analysis, manually assign relationships, and then expand them using knowledge base. Kontopoulos et al. [5] propose an ontology-based sentiment analysis method. They use pre-defined rules to extract entities and their attributes from

tweets, which are then used for matching entities in sentiment-related tweets. Their application however, is limited outside of product sentiment analysis.

3 Constructing Multiple Domain Taxonomy

Introduced in our previous work [15], an MDT is special ontology that has two types of relationships, namely, *domain association* and *taxonomy association*, denoted as *in_a* and *is_a*. *Domain associations* define the domain to which a concept belongs. *Taxonomy associations* define taxonomical hierarchies between concepts. In this section, we introduce a semi-automatic method for constructing MDTs. This construction method is based on semantic relativeness and aims to find concepts that have high discriminativeness for their domain.

3.1 Semantic Relation Strength

Throughout this work we use frequency-based semantic measures that do not depend on lexical patterns. The first measure is semantic relatedness, which we deploy in both MDT construction and domain classification. We use a measure of semantic relatedness called Normalized Google Distance (NGD), which was firstly proposed by Cilibrasi et al. [2]. It defines a semantic difference between two terms as:

$$NGD(a,b) = \frac{log(max(|A|,|B|)) - log(|A \bigcap B|)}{log(|W|) - log(min(|A|,|B|))}$$

where a and b are two terms, A and B are the sets of documents that contain a and b, respectively, and W is the entire document set.

The range of NGD(.) is between 0 and ∞, and a larger output indicates a lower semantic relatedness. However, if $|A|$ or $|B|$ is 0, the output is undefined. Following [4], we adopt NGD and make a function called Semantic Relation Strength (SRS) that is positively correlated to semantic relatedness:

$$SRS(a,b) = \begin{cases} -\infty & \text{if } |A|,\ |B| \text{ or } |A \bigcap B| \text{ is } 0 \\ 1 - NGD(a,b) & \text{otherwise} \end{cases}$$

3.2 Constructing MDT

In this work, we allow user input during MDT construction in two ways. First we require the user to provide some initial example concepts, for instance two to five concepts for each domain. Then in each construction iteration, the user will manually check and pick concepts from a ranked list of candidates to add to the MDT. The unsupervised tasks in this process include preparing candidate concepts generated from unlabeled data, and ranking candidates for selection. We will describe the technique below.

Preparing Candidate Concepts. Identifying name entities in unstructured texts by itself is a critical research topic. In our previous work [15], we follow many existing works and target single-word terms. However, in many cases a concept cannot be expressed using single-word terms. For example, "red cross" has a special meaning as a phrase, which cannot be expressed by "red" or "cross". So in this work, we target multi-word concepts.

We simplify the method proposed by Liu et al. [7], which aims to identify quality phrases based on concordance and informativeness measures. In their work, 200 to 300 labeled examples are required to provide satisfactory identification of quality phrases from text. In our simplified version, we do not require training examples, but use thresholds to select quality phrase candidates.

The technique starts by generating all n-gram phrases and their frequencies from text data, followed by calculating the concordance for each multi-word phrase. The concordance of a phrase is calculated using the *pointwise mutual information* (PMI) of the weakest separation of the phrase:

$$CON(v) = PMI(l, r) = \log \frac{f(v)}{f(l)f(r)}$$

where $f(.)$ is the frequency of a term, and $<l, r>$ is the separation of phrase v that provides the lowest PMI:

$$<l, r> \; = \; \arg\min_{l \oplus r = v} \log \frac{f(v)}{f(l)f(r)}$$

Since it is guaranteed that $f(l)f(r) \geq f(v)$, we will have $CON(v) \leq 0$. If v is a single-word term, we set $CON(v) = 0$.

Then Informativeness is calculated as the *inverse document frequency* (IDF):

$$INF(v) = IDF(v) = \log \frac{N}{|\{d \in D : v \in d\}|}$$

Generally a good candidate phrase v should have relatively high $CON(v)$ and $INF(v)$. We use thresholds to select candidates. From all generated phrases, the phrases that do not satisfy the threshold requirements are removed. The phrases that start and end with stopwords are also removed, following the pruning technique introduced in [7].

Ranking Discriminative Concepts. With one of the main applications being domain classification, an MDT should have concepts that have high discriminativeness for their respective domains. The discriminativeness of a concept for a domain shows how strong the concept is related to the specific domain but not to other domains. Here we propose a measure of discriminativeness of a term based on its semantic relatedness to concepts already defined in the MDT. Specifically, the discriminativeness of a term t for domain i can be calculated from:

$$DIS(t, i) = max(SRS(t, c), \forall c \in D_i) - \frac{1}{|D| - 1} \sum_{j \neq i} max(SRS(t, c), \forall c \in D_j)$$

where $\forall c \in D_i$ is all the concepts in domain i. We calculate this discriminativeness for all candidate terms in each iteration, and rank the top-k terms for manual checking.

4 Experimental Analysis

We test the effectiveness of our approach with experiments using real world short text data. While we have introduced a method to construct MDTs, it is difficult to directly estimate the quality of the taxonomy. We therefore determine the quality of a constructed MDT based on its effectiveness in solving text processing problems. In this section, we present our experiment on MDT-based short text classification. We describe datasets used, experimental setup, the process of iteratively constructing MDT, baseline methods and accuracy results in the following.

4.1 Datasets

Our experiments are conducted on two sets of real-world short text data from Twitter. The first dataset, called the *shooting* dataset, is collected using the Twitter Filter API[1] during September and October, 2014. The dataset has about 2 million tweets containing the keyword *shooting*. After removing retweets, we obtain a set of 284,343 tweets. We examine the data and discover that the tweets are mainly related to four domains, namely, $Crime$, $Imaging$, $Game$, and $Metaphor$. After deciding the domains, we label a number of tweets according to their domains. The labeled data contains 1,083 tweets, including 334 Crime, 245 Imaging, 257 Game, and 247 Metaphor tweets.

The second dataset is called the *crisis* dataset and is a publicly available dataset[2] introduced by Olteanu et al. [9]. It contains sets of tweets related to 26 natural disasters and other crisis events, including labeled and unlabeled tweets. Combining tweets for all 26 events and removing the retweets, we obtain 94,388 unlabeled tweets, and 3,674 labeled tweets. The labeled tweets contain five categories, namely, $Eyewitness$, $Business$, $Government$, $Media$, and NGO. The labels in this dataset are quite unbalanced. In labeled tweets, only 87 (2.35%) are Business, while 2,596 (70.6%) are Media. The first class, Eyewitness, has 528 items (14.3%) in the labeled data, and has relatively insignificant overlapping with other domains.

4.2 Experiment Settings and MDT Construction

We prepare the experiment by generating a list of concept candidates from unlabeled data, using the method describing in the previous section. Because the shooting dataset is a relatively larger dataset, we accordingly set the required

[1] https://dev.twitter.com/streaming/reference/post/statuses/filter.
[2] http://crisislex.org/.

support for a term to be considered as a concept candidate to a larger value. After applying the threshold, we obtain 6,670 candidate words and phrases for the shooting dataset, and 4,272 candidates for the crisis dataset.

Then we select some initial concepts from the candidate list, label their domain and taxonomy, and iteratively construct the MDT for both datasets. In each iteration, the DIS score per domain is calculated for all candidates, and an ordered list of top-100 candidates per domain are displayed for manual selection. From this list we pick a few terms, mostly from the top-10 candidates, to add to the MDT. We run five iterations for each dataset. Table 1 shows the number of concepts in MDT after each construction iteration.

Table 1. Number of concepts in MDT

Iteration	Initial	1	2	3	4	5
Shooting	21	32	41	47	59	68
Crisis	52	68	84	103	122	134

4.3 Domain Classification Results

We test the domain classification accuracy for our approach. The method for using MDT in domain classification can be found in our previous work [15]. We focus on the first domain for the two datasets, namely, *Crime* in the shooting dataset, and *Eyewitness* in the crisis dataset. We focus on these two domains because *Crime* and *Eyewitness* are more desirable information, and have been the topic in several studies [6,14]. Table 2 shows accuracy changes of these two domains over MDT construction iterations. In both cases, we see that the precision increases as more concepts being added to the MDT, while the recall drops, and the f1 measure stays about the same.

Table 2. Accuracy changes over MDT construction iteration

Iteration		Initial	1	2	3	4	5
Crime	Precision	0.72	0.73	0.74	0.77	0.79	0.80
	Recall	0.74	0.77	0.75	0.70	0.65	0.65
	f1	0.73	0.75	0.74	0.73	0.72	0.72
Eyewitness	Precision	0.53	0.54	0.59	0.61	0.63	0.67
	Recall	0.64	0.61	0.57	0.55	0.53	0.53
	f1	0.58	0.58	0.58	0.58	0.58	0.59

With the resulted MDT after five iterations, we compare our approach with four baselines. The first is *accept all* which considers all messages as positive. The *accept all* method would always achieve the highest recall of 1.0. The second

is the standard Bag-of-Words (BOW) method. For this method, we first generate a dictionary that contains terms appear over times than a frequency threshold k. For the shooting dataset, we use $k = 10$. For the crisis dataset, we use $k = 20$. The transformed tweets are then trained and tested using SVM classifier with three-fold cross validation. The third baseline, proposed by Sriram et al. [11], is a supervised method based on eight features and the Naive Bayes model. The forth baseline, based on the identification of *personal account* (PA), is from our previous work [14]. It is an unsupervised approach that incorporates lexical analysis and user profiling. This method is shown to be effective for filtering personal observations from tweet messages.

The classification accuracy, including precision, recall, and f1-value, of four baselines and our approach is shown in Table 3. As can be seen from the results, our approach achieved very high precision compared to the baselines. For classifying *crime* domain, it achieved 0.80 precision, which was a 16% increase from the baseline methods, as well as 0.72 f1-value, a 11% increase from the baseline methods. For classifying *eyewitness*, it also achieved a high precision of 0.67, a 3% increase from the baseline method. The PA method is designed to distinguish observation messages according to its source, and thus it achieved a low accuracy classifying *crime* which includes messages from various sources, but for *eyewitness* it achieved the highest accuracy among baseline methods. Our MDT-based method, nevertheless, surpassed the PA method both in precision and recall for classifying *eyewitness*.

Table 3. Classification accuracy of the first domains in two datasets

		Accept all	BOW	Sriram	PA	MDT
Crime	Precision	0.30	0.64	0.40	0.31	**0.80**
	Recall	**1**	0.59	0.71	0.49	0.65
	f1	0.46	0.61	0.51	0.38	**0.72**
Eyewitness	Precision	0.14	0.50	0.32	0.64	**0.67**
	Recall	**1**	0.50	0.52	0.50	0.53
	f1	0.24	0.50	0.40	0.56	**0.59**

5 Conclusion

In this paper, we present a method to semi-automatically construct *multiple domain taxonomy* (MDT) for processing multi-domain text data. The proposed semi-automatic construction method of MDT is based on unsupervised semantic analysis with unlabeled data, in combination with a small amount of manual effort in providing initial concepts and making selection from computer-provided candidates in each iteration. We have conducted extensive experiments on real-world Twitter datasets, which confirms the effectiveness of our approach in short text domain classification. The constructed MDT can achieve higher accuracy than existing methods in text domain classification. In the future, we plan to

investigate automatic update methods that can add emerging popular concepts to the taxonomy in realtime.

Acknowledgement. The research is partially supported by Natural Science Foundation of China (Nos. 61772005, 61300025) and Natural Science Foundation of Fujian Province (No. 2017J01753).

References

1. Castillo, C., Mendoza, M., Poblete, B.: Information credibility on Twitter. In: Proceedings of the 20th International World Wide Web Conference, pp. 675–684 (2011)
2. Cilibrasi, R.L., Vitanyi, P.M.: The Google similarity distance. IEEE Trans. Knowl. Data Eng. **19**(3), 370–383 (2007)
3. Dredze, M., Paul, M.J., Bergsma, S., Tran, H.: Carmen: a Twitter geolocation system with applications to public health. In: AAAI Workshop on Expanding the Boundaries of Health Informatics Using AI, pp. 20–24 (2013)
4. Han, X., Sun, L., Zhao, J.: Collective entity linking in web text: a graph-based method. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 765–774. ACM (2011)
5. Kontopoulos, E., Berberidis, C., Dergiades, T., Bassiliades, N.: Ontology-based sentiment analysis of Twitter posts. Expert Syst. Appl. **40**(10), 4065–4074 (2013)
6. Li, R., Lei, K.H., Khadiwala, R., Chang, K.-C.: TEDAS: a Twitter-based event detection and analysis system. In: Proceedings of 28th International Conference on Data Engineering, pp. 1273–1276 (2012)
7. Liu, J., Shang, J., Wang, C., Ren, X., Han, J.: Mining quality phrases from massive text corpora. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1729–1744. ACM (2015)
8. Lucia, W., Ferrari, E.: Egocentric: ego networks for knowledge-based short text classification. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management, pp. 1079–1088. ACM (2014)
9. Olteanu, A., Castillo, C., Diaz, F., Vieweg, S.: CrisisLex: a lexicon for collecting and filtering microblogged communications in crises. In: Proceedings of the 8th International AAAI Conference on Weblogs and Social Media, pp. 376–385 (2014)
10. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes Twitter users: real-time event detection by social sensors. In: Proceedings of the 19th International World Wide Web Conference, pp. 851–860 (2010)
11. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M.: Short text classification in twitter to improve information filtering. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 841–842 (2010)
12. Suliman, A.T., et al.: Event identification and assertion from social media using auto-extendable knowledge base. In: International Joint Conference on Neural Networks (IJCNN), pp. 4443–4450. IEEE (2016)
13. Unankard, S., Li, X., Sharaf, M., Zhong, J., Li, X.: Predicting elections from social networks based on sub-event detection and sentiment analysis. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y. (eds.) WISE 2014. LNCS, vol. 8787, pp. 1–16. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11746-1_1

14. Zhang, Y., Szabo, C., Sheng, Q.Z.: Improving object and event monitoring on Twitter through lexical analysis and user profiling. In: Cellary, W., Mokbel, M.F., Wang, J., Wang, H., Zhou, R., Zhang, Y. (eds.) WISE 2016. LNCS, vol. 10042, pp. 19–34. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48743-4_2
15. Zhang, Y., Szabo, C., Sheng, Q.Z., Zhang, W.E., Qin, Y.: Identifying domains and concepts in short texts via partial taxonomy and unlabeled data. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 127–143. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_9

Combining Bilingual Lexicons Extracted from Comparable Corpora: The Complementary Approach Between Word Embedding and Text Mining

Sourour Belhaj Rhouma[1(✉)], Chiraz Latiri[1(✉)], and Catherine Berrut[2(✉)]

[1] Faculty of Siences of Tunis, LIPAH-LR11ES14, University of Tunis El Manar, 2092 Tunis, Tunisia
sourour.bhr@gmail.com, chiraz.latiri@gnet.tn

[2] LIG Laboratory, MRIM Group, University of Grenoble Alpes, Grenoble, France
catherine.berrut@mrim.fr

Abstract. Recently, different works on bilingual lexicon extraction from comparable corpora have been proposed. This paper presents how to combine differents methods for bilingual lexicon extraction based on standard context vectors and advanced text mining methods. In this respect, we focus on combining bilingual lexicons based on context vectors, association rules and contextual meta-rules. The combination of lexicons leads to a less sparse representation in order to extract the most effective translations from these lexicons and create an optimal bilingual lexicon. An experimental validation conducted on two pairs of languages of the CLEF 2003 campaign evaluation, shows that the combination of the models give a significant improvement compared to the standard approach.

1 Introduction

Bilingual lexicons play a vital role in several applications related to Natural language Processing (NLP) and Information Retrieval (IR). In the last decade, some researches has been proposed to acquire Bilingual Lexicon Extraction (BLE) from comparable corpora [15]. Most of works in BLE tasks from comparable corpora represent the Standard Approach (SA) [1–3,16] which is based on the context vectors (CV). These vectors store a set of lexical terms which are for the neighbourhood of the base word and share the same lexical context. The relation between a base word and its context is called a co-occurrence relation. This approaches suffer from noisy vectors that affect their accuracy. Other approach for BLE based on co-occurence relation are proposed in [4]. These approaches rely on enriched representations of the word, especially those derived through Formal Concept Analysis (FCA) paradigm [5] and advanced Text Mining (TM) methods. These methods are deployed to extract Association Rules, introduced in [6], and extend them to contextual Meta-Rule (MR) [7]. These later capture all the words related to AR associated with the base word.

S. Hartmann et al. (Eds.): DEXA 2018, LNCS 11030, pp. 510–518, 2018.
https://doi.org/10.1007/978-3-319-98812-2_47

A proposal, which comes naturally to mind, consists in combining extracted lexicons in order to extract the most effective translations from the extracted lexicons and create an optimal bilingual lexicon. According to our knowledge, this is the first time for combining extracted bilingual lexicons based on CV with lexicons based on advanced TM methods. The objective is to improve the contextual representation and performance of the BLE from comparable corpora. The combination applied on extracted lexicons is handled through two strategies: one based on a union of all candidate translations and the second is based on a best-match combination. Thus, we concentrate on these two strategies to combine extracted lexicons and compare all extracted and combined lexicons with regard to that of the SA.

2 Related Works to Combining Bilingual Lexicons

Most of the works of the state of the art dealing with combining bilingual lexicons are based on using pivot language or linguistics resources. Most of approaches based on pivot language use dictionaries to translate into and from a pivot language in order to generate a new dictionary [8]. These pivot language based methods rely on the idea that searching for a word in an uncommon language could be executed through a third intermediated language. An interesting challenge in [9] proposed two novel combination models to combine different dictionaries together; existing dictionary and three dictionaries created with pivot-based schema considering three different languages as pivot. In other hand, cmbination approaches based on linguistics resources consist of combining lexicons extracted from multiple linguistics source knowledge such as: comparable corpora, bilingual thesaurus, preliminary existing dictionary, ... [10] presents a combination of two approaches (graphical and syntactic representations) by using the scores returned by each approach as a merge criterion. [11] merges the two lists by a simple arithmetic combination of scores. In addition, other methods of score combinations have been tested as the harmonic combination of ranks and scores.

Our contribution is considered as a linguistics resources approach for combination of extracted bilingual lexicons from comparable corpora. Most of works which combine methods for BLE use a classic way to merge methods by taking, as input, the list of scores or ranks of each of the methods. In our case, for each word to translate, we take as input a list of candidate translations returned by each method. Therefore, it is very important to combine the bilingual lexicon based on CV with those based on AR and MR having improvements alone [4].

3 Bilingual Lexicon Extraction from Comparable Corpora

3.1 BLE Based on Context Vector: Standard Approach

Most of the works dealing with BLE tasks from comparable corpora are based on the SA [1–3]. The main assumption of the SA states that words with a similar

meaning are likely to appear in similar context across languages. Therefore, a word can be represented as a context vector (CV) in source or target language. In order to enable the comparison of source and target CV, words in the source CV are translated into the target language using an initial bilingual dictionary (IBD). The use of an IBD to translate CV is justified by the fact that some of the elements of the CV are likely to belong to the general language and to the bilingual dictionary. Thus, we can expect the translated CV of a word in source language w_S to be closer to the CV of the translation of w_S. In a final step, candidate translations are being ranked according to a distance measure.

Let BL_{CV} is the bilingual lexicon based on context vectors such as:

$$BL_{CV} = \{(w_S, TL_{CV}(w_S)\} \; where \; TL_{CV}(w_S) = \{w_{T1}, \ldots, w_{Tl}\} \qquad (1)$$

$TL_{CV}(w)$ is the translation list correesponding to w and $w_{T1}, \ldots, w_{Tl}$ are candidate translations corresponding to w in the bilingual lexicon extracted with the SA.

3.2 BLE Based on Advanced Text Mining Methods

Our goal is to extract bilingual lexicons using two patterns association rules (AR) and contextual meta-rules (MR), providing additional and implicit knowledge. The implementation of this approach can be carried out by applying in three major steps [4]:

1. Extraction of two sets of source and target AR/MR generated from a comparable corpora: An AR approximates the probability of having the terms of the conclusion in a document, given that those of the premise are already there. A MR provides a global context for the terms that appear together and related to AR associated with the same base word (premise).
2. Translation to target language of the conclusion of AR/MR by using a bilingual resource.
3. Each translated AR or MR is compared to the sets of AR or MR intarget language to filter the most similar ones by using similarity measures.
4. Construction of bilingual lexicon by associating each AR/MR in source language with the most similar AR/MR in target language.

Let denote that BL_{AR} and BL_{MR} are bilingual lexicons based on association rules and meta-rules, respectively:

$$BL_{AR} = \{(w_S, TL_{AR}(w_S))\} \; where \; TL_{AR}(w_S) = \{w_{T1}, \ldots, w_{Ti}\} \qquad (2)$$

$$BL_{MR} = \{(w_S, TL_{MR}(w_S))\} \; where \; TL_{MR}(w_S) = \{w_{T1}, \ldots, w_{Tj}\} \qquad (3)$$

$TL_{AR}(w)$ and $TL_{MR}(w)$ are translation lists corresponding to w. $w_{T1}, \ldots, w_{Ti}$ and $w_{T1}, \ldots, w_{Tj}$ are candidate translations corresponding to w, for extracted bilingual lexicons based on AR and MR.

4 Combining Bilingual Lexicons Extracted from Comparable Corpora

The combination applied on extracted lexicons is handled through two strategies: one based on a union of all candidate translations and the second is based on a best-match strategy.

4.1 Using the Union Combination

The union combination strategy can be directly applied to the extracted lexicons by integrating all candidate translations given for each souce word w_S. Our union combination rule is as follows:

Suppose that translation lists for each lexicon are already sorted according to a similarity score, if there is the source word w_S in more than one lexicon, their corresponding translations for the combined bilingual lexicon is the union of translation lists from the lexicons by respcting the order.

By applying the above union rule, a new combined lexicon is created, which is denoted BL_{CbU}. Moreover, to build the new lexicon, for each word w_S we scroll vertically through the translation lists given by each lexicon to first process the translations with the best similarity score. If a source word w_S appears in only one lexicon, the corresponding translation list will be included in the combined lexicon for the word w_S. The union combination function is as follows:

$$BL_{CbU} = \cup\{(w_S, TL_{CbU}(w_S))\} \tag{4}$$

where $TL_{CbU}(w)$ is the union of the set of the translation lists given by each lexicon for w in the combined bilingual lexicon.

$$TL_{CbU}(w) = TL_{CV}(w) \cup TL_{AR}(w) \cup TL_{MR}(w) \tag{5}$$

In the case where the combination is between two extracted lexicons, $TL_{CbU}(w)$ is the translation list corresponding to w, and $TL_{CbU}(w) = TL_{CV} \cup TL_{AR}$ or $TL_{CbU}(w) = TL_{CV} \cup TL_{MR}$ which is the union of two translation lists corresponding to w.

4.2 Using the Best-Match Combination

Our best-match combination rule is:

if there is the source word w_S in more than one lexicon, suppose that one of the lexicon have the best translation, their corresponding translations are selected for the combined bilingual lexicon.

By applying the Best-Match combination, a new lexicon denoted by BL_{cbBM} is created. For each entry, we select the corresponding translations of w_S in the lexicon where the correct translation have the best rank regards to the intial

bilingual dictionary IBD already used for translation step in Subsects. 3.1 and 3.2. The best-match combination function is as follows:

$$BL_{CbBM} = \{(w_S, TL_{CbBM}(w_S))\} \tag{6}$$

where TL_{CbBM} is the selected translation list for w_S in the combined bilingual lexicon. This means for each w, we select $TL_{CbBM}(w)$ from the set of the translations lists given by each lexicon. The best-match translation list $TL_{CbBM}(w)$ is defined as follows:

$$TL_{CbBM}(w) = BMatch\{TL_{CV}(w), TL_{AR}(w), TL_{MR}(w)\} \tag{7}$$

In the case where the combination is between two extracted lexicons, $TL_{CbBM}(w) = BMatch\{TL_{CV}(w), TL_{AR}(w)\}$ or $TL_{CbBM}(w) = BMatch\{TL_{CV}(w), TL_{MR}(w)\}$ which is the combination of two translation lists. The best translation list TL_{CbBM} is carried out by the best match function $BMatch$ applied for each translation list. This function is based on the rank score R_i computed for each translation list i, defined as follows:

$$R_i = \sum_{t=1}^{LE_i} \frac{1}{CR_t} \tag{8}$$

where LE_i is the length of the translation list i and CR_t is the rank of the correct translation t regards to the IBD dictionary. If the correct translation does not appear in the translation list, $\frac{1}{CR_t}$ is set to 0.

5 Experimental Validation

5.1 Corpora and Linguistic Resources

We evaluate our BLE approaches on *CLEF 2003* presented in Table 1, we rely on a standard journalistic collection used in CLIR field. We conducted experiments on French-English and italian-English collections proposed in the multilingual track of CLEF2003. The translation is handled usin IBD which is the linguistic resource BabelNet[1] [12] due to its lexicographic and encyclopedic coverage of multilingual terms resulting from a mapping of the Wikipedia pages[2] and WordNet, with other lexical semantic resources.

5.2 Evaluation

To evaluate the quality of our approaches regards to SA, we built a bilingual reference list. The method of creating the reference list differs regards to previous works. For CLEF 2003, we created two reference lists for the CLEF 2003

[1] babelnet.org.
[2] http://www.wikipedia.org.

(FR-EN) and (IT-EN). We selected 400 French words with their corresponding translations in English from the online dictionary *Word Reference* [3]. The second list is composed of 120 italian single terms with their translations extracted from the same source. The rank of the correct translation can be considered as an important characteristic when evaluating extracted lexicons. This characteristic is taken into account only by measuring the mean average precision (MAP) defined in [3]. This measure allows to present the capacity of the method to order exactly selected translation candidates. Let n is the number of terms of the reference list, N is the length of candidates translation and r_i is the rank of the correct candidate translation i. If the correct translation does not appear in the top N candidates, $\frac{1}{r_i}$ is set to 0. The MAP is defined as follows:

$$MAP = \frac{1}{n} \sum_{i=1}^{N} \frac{1}{r_i} \tag{9}$$

5.3 Evaluation Scenarios

We carried out different evaluation scenarios from collections of CLEF 2003 campaign for the two pairs of languages: French-English and Italian-Eglish. The baseline is the application of the standard approach based on context vectors, denoted by BL_{CV}, on the same collections and the same resources used by the scenarios.

1. **BLE based on association rule**: denoted by BL_{AR}, consist to evaluate the extracted bilingual lexicon based on AR by referring to a reference list.
2. **BLE based on contextual meta-rule**: denoted by BL_{MR}, consist to evaluate the extracted bilingual lexicon based on MR by referring to a reference list.
3. **Combined bilingual lexicons using union combination strategy**: denoted by BL_{CbU}. This combination strategy can be used to combine two or three lexicons, as follows:
   - $BL_{CbU-CV+AR}$ and $BL_{CbU-CV+MR}$: which use the union combination strategy to combine BL_{CV} with BL_{AR} and respectively BL_{CV} with BL_{MR}.
   - $BL_{CbU-CV+AR+M}$: which apply the union combination strategy on the three extracted lexicons based on CV, MR and AR.
4. **Combined bilingual lexicons using best-match combination strategy**: denoted by BL_{CbBM}. This combination strategy can be used to combine two or three lexicons, as follows:
   - $BL_{CbBM-CV+AR}$ and $BL_{CbBM-CV+MR}$: which use the best-match combination strategy to combine BL_{CV} with BL_{AR} and respectively BL_{CV} with BL_{MR}.
   - $BL_{CbBM-CV+MR+AR}$: which apply the best-match combination strategy on the three extracted lexicons based on CV, MR and AR.

[3] http://wordreference.com.

5.4 Results and Discussion

Table 1 shows obtained results in terms of MAP. We interpret that our approach based on meta-rules BL_{MR} were improve significantly for the three corpora compared to BL_{CV}. This can be explained by the noisy nature of the context vectors with respect to the filtered meta-rules by setting thresholds of support and confidence for their construction. The BL_{MR} is better in terms of MAP compared to BL_{AR}, and this can be explained by the fact that the rank of the correct translations found for the lexicon based on MR is more important than the correct translations found for the lexicon based on AR. For combined lexicons, we demonstrate that approaches which combines BL_{CV} with BL_{MR} or BL_{MR} ($BL_{CbU-CV+AR}$, $BL_{CbU-CV+AR}$, $BL_{CbBM-CV+AR}$ and $BL_{CbBM-CV+AR}$) were improved significantly more than BL_{MR} and bL_{AR} by using the two combination strategies resulting BL_{CombU} and BL_{CombBM}. Besides, approches which integrates meta-rules, association rules and context vectors ($BL_{CbU-CV+MR+AR}$ and $BL_{CbBM-CV+MR+AR}$) gives the highly significant difference ($p < 0.01$) for the two combination strategies, for CLEF 2003 (FR-EN) with p equal to 0.0082 and for CLEF 2003 (IT-EN) with a p equal to 0.0074). All experiments for the two pairs of languages shows that the BL_{CbBM} yields better results than BL_{CbU} for the combination of two or three lexicons. The obtained results for combined lexicons $BL_{CbBM-CV+MR+AR}$ in term of MAP (19.35% by using the union combination and 19.55% by using the best-match combination for CLEF 2003 (FR-EN)) are better than the ones reported in [13] (17.05% by the greedy appoach) and those obtained in [14] (17.50% for the weighted combination approach which integrates concept vectors with context vectors). Furtheremore, the quality of combined lexicons$BL_{CbBM-CV+MR+AR}$ with the best-match combination strategy is highly significant. This improvement is explained by the fact that selected translation list for each source word possesses correct translations in the best rank.

Table 1. MAP improvement achieved for extracted lexicons and combined lexicons with the two combination strategies. The symbols † indicates statistically significant improvement over the best run in bold, $p - value < 0.01$

Extracted lexicons	MAP	Union combination	MAP	BMatch combination	MAP
CLEF 2003 (FR-EN)					
BL_{CV}	0.1610	$BL_{CbU-CV+AR}$	0.1696	$BL_{CbBM-CV+AR}$	0.1702
BL_{AR}	0.1669	$BL_{CbU-CV+MR}$	**0.1920**†	$BL_{CbBM-CV+MR}$	**0.1932**†
BL_{MR}	0.1902	$BL_{CbU-CV+AR+MR}$	**0.1935**†	$BL_{CbBM-CV+AR+MR}$	**0.1955**†
CLEF 2003 (IT-EN)					
BL_{CV}	0.201	$BL_{CbU-CV+AR}$	0.2232	$BL_{CbBM-CV+AR}$	0.2239
BL_{AR}	0.2135	$BL_{CbU-CV+MR}$	**0.2644**†	$BL_{CbBM-CV+MR}$	**0.2655**†
BL_{MR}	0.2277	$BL_{CbU-CV+AR+MR}$	**0.1935**†	$BL_{CbBM-CV+AR+MR}$	**0.1955**†

6 Conclusion and Perspectives

In this article, we are interested in two main ways for BLE from comparable corpora, namely: bilingual lexicon based on standard approach as well as bilingual lexicons based on TM methods. We then introduced two new strategies for combining these extracted bilingual lexicons. The two strategies of proposed contextual combinations showed results superior to the use of each representation separately. We evaluated our approaches on a general corpora made of news articles in two pairs of language. More precisely, our different experiments show that using the two proposed combination strategies always improves significantly the quality of extracted lexicons in term of MAP. Furtheremore, the quality of combined lexicons using context vectors, association rules and meta-rules with the best-match combination strategy is highly significant. We hope that this work will pave the way for further research concerning to improve the quality of the extracted lexicons. Secondly, we can propose a CLIR model based on the extracted and combined lexicons.

References

1. Rapp, R.: Automatic identification of word translations from unrelated English and German corpora. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, pp. 519–526 (1999)
2. Morin, E., Daille, B., Takeuchi, K., Kageura, K.: Bilingual terminology mining-using brain, not brawn comparable corpora. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007), Prague, Czech Republic, pp. 664–671 (2007)
3. Morin, E., Hazem, A.: Exploiting unbalanced specialized comparable corpora for bilingual lexicon extraction. Nat. Lang. Eng. **22**(4), 575–601 (2016)
4. Belhaj Rhouma, S., Latiri, C., Catherine, B.: Advanced text mining methods for bilingual lexicon extraction from specialized comparable corpora. In: 19th International Conference on Computational Linguistics and Intelligent Text Processing, Hanoi, Vietnam. Lecture Notes in Computer Science. Springer, 18–24 March 2018
5. Ganter, B., Wille, R.: Formal Concept Analysis. Springer, Berlin (1999). https://doi.org/10.1007/978-3-642-59830-2
6. Agrawal, R., Skirant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Databases, VLDB 1994, Santiago, Chile, pp. 478–499, September 1994
7. Sourour, B.R., Latiri, C.C., Slimani, Y.: Vers des méta-règles de contexte appréciées par la IIE pour la RI. In: CORIA 2015 - Conférence en Recherche d'Infomations et Applications - 12th French Information Retrieval Conference, Paris, France, pp. 205–220, 18–20 March 2015
8. Tanaka, K., Umemura, K.: Construction of a bilingual dictionary intermediated by a third language. In: Proceedings of the 15th Conference on Computational Linguistics, Kyoto, Japan, vol. 1, pp. 297–303. Association for Computational Linguistics (1994)
9. Ansari, E., Sadreddini, M.H., Alireza, T., Sheikhalishahi, M.: Combining different seed dictionaries to extract lexicon from comparable corpus. Indian J. Sci. Technol. **7**(9), 1279–1288 (2014)

10. Hazem, A., Morin, E.: Bilingual lexicon extraction from comparable corpora by combining contextual representations. In: Traitement Automatique des Langues Naturelles, TALN 2013, pp. 243–256, 17–21 Juin 2013. Articles longs
11. Prochasson, E., Morin, E.: Points d'ancrage pour l'extraction lexicale bilingue à partir de petits corpus comparables spécialisés. In: Traitement Automatique des Langues (TAL), vol. 50, pp. 283–304 (2009)
12. Navigli, R., Ponzetto, S.: Babelnet: building a very large multilingual semantic network. In: Hajic, J., Carberry, S., Clark, S. (eds.) Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010), pp. 216–225. The Association for Computer Linguistics (2010)
13. Li, B., Gaussier, E.: Improving corpus comparability for bilingual lexicon extraction from comparable corpora. In: 23rd International Conference on Computational Linguistics, Proceedings of the Conference (COLING 2010), pp. 644–652, 23–27 August 2010
14. Chebel, M., Latiri, C., Gaussier, E.: Bilingual lexicon extraction from comparable corpora based on closed concepts mining. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 586–598. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57454-7_46
15. Hazem, A., Morin, E.: Efficient data selection for bilingual terminology extraction from comparable corpora. In: 26th International Conference on Computational Linguistics (COLING 2016), pp. 3401–3411. ACL, Osaka, Japan (2016)
16. Bouamor, D., Semmar, N., Zweigenbaum, P.: Towards a generic approach for bilingual lexicon extraction from comparable corpora. In: Proceedings of the 15th Machine Translation Summit. pp. 143–150, 2–6 Sept 2013

Author Index