Hendrik Decker
Lenka Lhotská
Sebastian Link
Josef Basl
A Min Tjoa (Eds.)

LNCS 8056

Database and Expert Systems Applications

24th International Conference, DEXA 2013
Prague, Czech Republic, August 2013
Proceedings, Part II

2 Part II

DEXA 2013

Springer

Lecture Notes in Computer Science 8056

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

Hendrik Decker Lenka Lhotská
Sebastian Link Josef Basl A Min Tjoa (Eds.)

Database and Expert Systems Applications

24th International Conference, DEXA 2013
Prague, Czech Republic, August 26-29, 2013
Proceedings, Part II

Springer

Volume Editors

Hendrik Decker
Instituto Tecnológico de Informática, Valencia, Spain
E-mail: hendrik@iti.es

Lenka Lhotská
Czech Technical University in Prague, Czech Republic
E-mail: lhotska@fel.cvut.cz

Sebastian Link
The University of Auckland, New Zealand
E-mail: s.link@auckland.ac.nz

Josef Basl
University of Economics, Prague, Czech Republic
E-mail: basl@vse.cz

A Min Tjoa
Vienna University of Technology, Austria
E-mail: amin@ifs.tuwien.ac.at

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-40172-5 e-ISBN 978-3-642-40173-2
DOI 10.1007/978-3-642-40173-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013944804

CR Subject Classification (1998): H.2, H.3, H.4, I.2, H.5, J.1, C.2

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

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

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

Preface

The book you are reading comprises the research articles as well as the abstracts of invited talks presented at DEXA 2013, the 24th International Conference on Database and Expert Systems Applications. The conference was held in Prague, the lovely Czech capital, where DEXA already took place in 1993 and 2003. The presented papers show that DEXA has successfully stayed true to a core of themes in the areas of databases, intelligent systems, and related applications, but also that DEXA promotes changing paradigms, new developments, and emerging trends. For the 2013 edition, we had called for novel results or qualified surveys in a wide range of topics, including:

* Acquisition, Modeling, Management and Processing of Knowledge
* Authenticity, Consistency, Integrity, Privacy, Quality, Security of Data
* Availability
* Constraint Modeling and Processing
* Database Federation and Integration, Interoperability, Multi-Databases
* Data and Information Networks
* Data and Information Semantics
* Data and Information Streams
* Data Provenance
* Data Structures and Data Management Algorithms
* Database and Information System Architecture and Performance
* Data Mining and Data Warehousing
* Datalog 2.0
* Decision Support Systems and Their Applications
* Dependability, Reliability and Fault Tolerance
* Digital Libraries
* Distributed, Parallel, P2P, Grid, and Cloud Databases
* Incomplete and Uncertain Data
* Inconsistency Tolerance
* Information Retrieval
* Information and Database Systems and Their Applications
* Metadata Management
* Mobile, Pervasive and Ubiquitous Data
* Modeling, Automation and Optimization of Processes
* Multimedia Databases
* NoSQL and NewSQL Databases
* Object, Object-Relational, and Deductive Databases
* Provenance of Data and Information
* Replicated Databases
* Semantic Web and Ontologies
* Sensor Data Management

* Statistical and Scientific Databases
* Temporal, Spatial, and High-Dimensional Databases
* User Interfaces to Databases and Information Systems
* WWW and Databases, Web Services
* Workflow Management and Databases
* XML and Semi-structured Data

In response to this call, we received 174 submissions from all over the world, of which 43 are included in these proceedings as accepted full papers, and 33 as short papers. We are grateful to the many authors who submitted their work to DEXA. Decisions on acceptance or rejection were based on at least three reviews for each submission. Most of the reviews were detailed and provided constructive feedback to the authors. We owe our deepest thanks to all members of the Program Committee and to the external reviewers who invested their expertise, interest, and time in their reviews.

The program of DEXA 2013 was enriched by three exceptional invited keynote speeches, presented by distinguished colleagues:

- Trevor Bench-Capon: "Structuring E-Participation in Policy Making Through Argumentation"
- Tova Milo: "Making Collective Wisdom Wiser"
- Klaus-Dieter Schewe: "Horizontal and Vertical (Business Process) Model Integration"

In addition to the main conference track, DEXA 2013 also featured seven workshops that explored a wide spectrum of specialized topics of growing general importance. The organization of the workshops was chaired by Franck Morvan, A. Min Tjoa, and Roland R. Wagner, to whom we say "many thanks indeed" for their smooth and effective work.

Special thanks go to the host of DEXA 2013, the Prague University of Economics, where, under the admirable guidance of the DEXA 2013 General Chairs Josef Basl and A. Min Tjoa, an excellent working atmosphere was provided.

Last, but not at all least, we express our heartfelt gratitude to Gabriela Wagner. Her professional attention to detail, skillful management of the DEXA event as well as her preparation of the proceedings volumes are greatly appreciated.

August 2013

Hendrik Decker
Lenka Lhotská
Sebastian Link

Organization

Honorary Chair

Makoto Takizawa	Seikei University, Japan

General Chairs

Josef Basl	University of Economics, Prague, Czech Republic
A. Min Tjoa	Technical University of Vienna, Austria

Conference Program Chairs

Hendrik Decker	Instituto Tecnológico de Informática, Valencia, Spain
Lenka Lhotská	Czech Technical University, Czech Republic
Sebastian Link	The University of Auckland, New Zealand

Publication Chair

Vladimir Marik	Czech Technical University, Czech Republic

Workshop Chairs

Franck Morvan	IRIT, Paul Sabatier University, Toulouse, France
A. Min Tjoa	Technical University of Vienna, Austria
Roland R. Wagner	FAW, University of Linz, Austria

Program Committee

Slim Abdennadher	German University, Cairo, Egypt
Witold Abramowicz	The Poznan University of Economics, Poland
Hamideh Afsarmanesh	University of Amsterdam, The Netherlands
Riccardo Albertoni	Italian National Council of Research, Italy
Rachid Anane	Coventry University, UK
Annalisa Appice	Università degli Studi di Bari, Italy
Mustafa Atay	Winston-Salem State University, USA
Spiridon Bakiras	City University of New York, USA

Zhifeng Bao	National University of Singapore, Singapore
Ladjel Bellatreche	ENSMA, France
Nadia Bennani	INSA Lyon, France
Morad Benyoucef	University of Ottawa, Canada
Catherine Berrut	Grenoble University, France
Debmalya Biswas	Nokia Research, Switzerland
Athman Bouguettaya	RMIT, Australia
Danielle Boulanger	MODEME, University of Lyon, France
Omar Boussaid	University of Lyon, France
Stephane Bressan	National University of Singapore, Singapore
Patrick Brezillon	University Paris VI (UPMC), France
Yingyi Bu	University of California, Irvine, USA
Luis M. Camarinha-Matos	Universidade Nova de Lisboa + Uninova, Portugal
Yiwei Cao	RWTH Aachen University, Germany
Silvana Castano	Università degli Studi di Milano, Italy
Barbara Catania	Università di Genova, Italy
Michelangelo Ceci	University of Bari, Italy
Wojciech Cellary	Poznan University of Economics, Poland
Cindy Chen	University of Massachusetts Lowell, USA
Phoebe Chen	La Trobe University, Australia
Shu-Ching Chen	Florida International University, USA
Hao Cheng	Yahoo
Reynold Cheng	The University of Hong Kong, Hong Kong
Max Chevalier	IRIT - SIG, Université de Toulouse, France
Byron Choi	Hong Kong Baptist University, Hong Kong
Henning Christiansen	Roskilde University, Denmark
Soon Ae Chun	City University of New York, USA
Eliseo Clementini	University of L'Aquila, Italy
Gao Cong	Microsoft Research Asia, UK
Oscar Corcho	Universidad Politécnica de Madrid, Spain
Bin Cui	Peking University, China
Deborah Dahl	Conversational Technologies
Jérôme Darmont	Université de Lyon (ERIC Lyon 2), France
Andre de Carvalho	University of Sao Paulo, Brazil
Guy De Tré	Ghent University, Belgium
Olga De Troyer	Vrije Universiteit Brussel, Belgium
Roberto De Virgilio	Università Roma Tre, Italy
John Debenham	University of Technology, Sydney, Australia
Hendrik Decker	Instituto Tecnológico de Informática, Valencia, Spain
Zhi-Hong Deng	Peking University, China
Vincenzo Deufemia	Università degli Studi di Salerno, Italy
Claudia Diamantini	Università Politecnica delle Marche, Italy
Juliette Dibie-Barthélemy	AgroParisTech, France

Ying Ding	Indiana University, USA
Zhiming Ding	Institute of Software, Chinese Academy of Sciences, China
Gillian Dobbie	University of Auckland, New Zealand
Peter Dolog	Aalborg University, Denmark
Dejing Dou	University of Oregon, USA
Cedric du Mouza	CNAM, France
Johann Eder	University of Klagenfurt, Austria
Suzanne Embury	The University of Manchester, UK
Bettina Fazzinga	University of Calabria, Italy
Leonidas Fegaras	The University of Texas at Arlington, USA
Victor Felea	"Al.I. Cuza" University of Iasi, Romania
Stefano Ferilli	University of Bari, Italy
Flavio Ferrarotti	Victoria University of Wellington, New Zealand
Filomena Ferrucci	Università di Salerno, Italy
Flavius Frasincar	Erasmus University Rotterdam, The Netherlands
Bernhard Freudenthaler	Software Competence Center Hagenberg GmbH, Austria
Hiroaki Fukuda	Shibaura Institute of Technology, Japan
Steven Furnell	Plymouth University, UK
Aryya Gangopadhyay	University of Maryland Baltimore County, USA
Yunjun Gao	Zhejiang University, China
Manolis Gergatsoulis	Ionian University, Greece
Bernard Grabot	LGP-ENIT, France
Fabio Grandi	University of Bologna, Italy
Carmine Gravino	University of Salerno, Italy
Sven Groppe	Lübeck University, Germany
William Grosky	University of Michigan, USA
Jerzy Grzymala-Busse	University of Kansas, USA
Francesco Guerra	Università degli Studi Di Modena e Reggio Emilia, Italy
Giovanna Guerrini	University of Genova, Italy
Antonella Guzzo	University of Calabria, Italy
Abdelkader Hameurlain	Paul Sabatier University, France
Ibrahim Hamidah	Universiti Putra Malaysia, Malaysia
Wook-Shin Han	Kyungpook National University, Republic of Korea
Takahiro Hara	Osaka University, Japan
André Hernich	Humboldt-Universität zu Berlin, Germany
Francisco Herrera	University of Granada, Spain
Steven Hoi	Nanyang Technological University, Singapore
Estevam Rafael Hruschka Jr.	Federal University of Sao Carlos, Brazil
Wynne Hsu	National University of Singapore, Singapore

Yu Hua	Huazhong University of Science and Technology, China
Jimmy Huang	York University, Canada
Xiaoyu Huang	South China University of Technology, China
Michal Huptych	Czech Technical University in Prague, Czech Republic
San-Yih Hwang	National Sun Yat-Sen University, China (Taiwan Province)
Theo Härder	TU Kaiserslautern, Germany
Ionut Emil Iacob	Georgia Southern University, USA
Sergio Ilarri	University of Zaragoza, Spain
Abdessamad Imine	University of Nancy, France
Yasunori Ishihara	Osaka University, Japan
Adam Jatowt	Kyoto 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
Sang-Wook Kim	Hanyang University, Republic of Korea
Benny Kimelfeld	IBM Almaden, USA
Hiroyuki Kitagawa	University of Tsukuba, Japan
Carsten Kleiner	University of Applied Sciences and Arts Hannover, Germany
Solmaz Kolahi	Oracle, USA
Ibrahim Korpeoglu	Bilkent University, Turkey
Harald Kosch	University of Passau, Germany
Michal Krátký	Technical University of Ostrava, Czech Republic
Petr Kremen	Czech Technical University in Prague, Czech Republic
Arun Kumar	IBM Research - India, India
Ashish Kundu	IBM T J Watson Research Center, Yorktown Heights, USA
Josef Küng	University of Linz, Austria
Kwok-Wa Lam	University of Hong Kong, Hong Kong
Nadira Lammari	CNAM, France
Gianfranco Lamperti	University of Brescia, Italy
Anne Laurent	LIRMM, University of Montpellier 2, France
Mong Li Lee	National University of Singapore, Singapore
Alain Léger	FT R&D Orange Labs Rennes, France
Daniel Lemire	LICEF, Université du Québec, Canada
Lenka Lhotská	Czech Technical University, Czech Republic
Wenxin Liang	Dalian University of Technology, China
Stephen W. Liddle	Brigham Young University, USA
Lipyeow Lim	Unversity of Hawaii at Manoa, USA

Tok Wang Ling	National University of Singapore, Singapore
Sebastian Link	The University of Auckland, New Zealand
Volker Linnemann	University of Lübeck, Germany
Chengfei Liu	Swinburne University of Technology, Australia
Chuan-Ming Liu	National Taipei University of Technology, China (Taiwan Province)
Fuyu Liu	Microsoft Corporation, USA
Hong-Cheu Liu	University of South Australia, Australia
Hua Liu	Xerox Research Labs at Webster, USA
Jorge Lloret Gazo	University of Zaragoza, Spain
Peri Loucopoulos	Harokopio University of Athens, Greece
Jiaheng Lu	Renmin University of China, China
Jianguo Lu	University of Windsor, Canada
Alessandra Lumini	University of Bologna, Italy
Hui Ma	Victoria University of Wellington, New Zealand
Qiang Ma	Kyoto University, Japan
Stéphane Maag	TELECOM SudParis, France
Elio Masciari	ICAR-CNR, Università della Calabria, Italy
Norman May	SAP AG, Germany
Jose-Norberto Mazón	University of Alicante, Spain
Dennis McLeod	University of Southern California, USA
Brahim Medjahed	University of Michigan - Dearborn, USA
Alok Mishra	Atilim University, Ankara, Turkey
Harekrishna Mishra	Institute of Rural Management Anand, India
Sanjay Misra	University of Technology, Minna, Nigeria
Jose Mocito	INESC-ID/FCUL, Portugal
Lars Moench	University of Hagen, Germany
Riad Mokadem	IRIT, Paul Sabatier University, France
Anirban Mondal	University of Tokyo, Japan
Yang-Sae Moon	Kangwon National University, Republic of Korea
Reagan Moore	University of North Carolina at Chapel Hill, USA
Franck Morvan	IRIT, Paul Sabatier University, France
Mirco Musolesi	University of Birmingham, UK
Tadashi Nakano	University of California, Irvine, USA
Ullas Nambiar	IBM Research
Ismael Navas-Delgado	University of Málaga, Spain
Martin Necasky	Charles University in Prague, Czech Republic
Wilfred Ng	University of Science and Technology, Hong Kong
Javier Nieves Acedo	University of Deusto, Spain

Levent V. Orman	Cornell University, Ithaca, New York, USA
Mourad Oussalah	University of Nantes, France
Gultekin Ozsoyoglu	Case Western Reserve University, USA
George Pallis	University of Cyprus, Cyprus
Christos Papatheodorou	Ionian University and "Athena" Research Centre, Greece
Marcin Paprzycki	Polish Academy of Sciences, Warsaw Management Academy, Poland
Óscar Pastor López	Universidad Politécnica de Valencia, Spain
Dhaval Patel	National University of Singapore, Singapore, Singapore
Jovan Pehcevski	European University, Macedonia, Former Yugoslav Republic
Jorge Perez	Universidad de Chile, Chile
Reinhard Pichler	Technische Universität Wien, Austria
Olivier Pivert	Ecole Nationale Supérieure des Sciences Appliquées et de Technologie, France
Clara Pizzuti	Institute for High Performance Computing and Networking (ICAR)-National Research Council (CNR), Italy
Jaroslav Pokorny	Charles University in Prague, Czech Republic
Pascal Poncelet	LIRMM, France
Elaheh Pourabbas	National Research Council, Italy
Xiaojun Qi	Utah State University, USA
Fausto Rabitti	ISTI, CNR Pisa, Italy
Claudia Raibulet	Università degli Studi di Milano-Bicocca, Italy
Isidro Ramos	Technical University of Valencia, Spain
Praveen Rao	University of Missouri-Kansas City, USA
Rodolfo F. Resende	Federal University of Minas Gerais, Brazil
Claudia Roncancio	Grenoble University/LIG, France
Edna Ruckhaus	Universidad Simon Bolivar, Venezuela
Massimo Ruffolo	ICAR-CNR, Italy
Igor Ruiz-Agundez	University of Deusto, Spain
Giovanni Maria Sacco	University of Turin, Italy
Shazia Sadiq	The University of Queensland, Australia
Simonas Saltenis	Aalborg University, Denmark
Carlo Sansone	Università di Napoli "Federico II", Italy
Igor Santos Grueiro	Deusto University, Spain
Ismael Sanz	Universitat Jaume I, Spain
N.L. Sarda	I.I.T. Bombay, India
Marinette Savonnet	University of Burgundy, France
Raimondo Schettini	Università degli Studi di Milano-Bicocca, Italy
Peter Scheuermann	Northwestern University, USA

Klaus-Dieter Schewe	Software Competence Centre Hagenberg, Austria
Erich Schweighofer	University of Vienna, Austria
Florence Sedes	IRIT, Paul Sabatier University, Toulouse, France
Nazha Selmaoui	University of New Caledonia, New Caledonia
Patrick Siarry	Université Paris 12 (LiSSi), France
Gheorghe Cosmin Silaghi	Babes-Bolyai University of Cluj-Napoca, Romania
Leonid Sokolinsky	South Ural State University, Russian Federation
Bala Srinivasan	Monash University, Australia
Umberto Straccia	Italian National Research Council, Italy
Darijus Strasunskas	DS Applied Science, Norway
Lena Strömbäck	Swedish Meteorological and Hydrological Institute, Sweden
Aixin Sun	Nanyang Technological University, Singapore
Raj Sunderraman	Georgia State University, USA
David Taniar	Monash University, Australia
Cui Tao	Mayo Clinic, USA
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
Taro Tezuka	University of Tsukuba, Japan
Bernhard Thalheim	Christian Albrechts Universität Kiel, Germany
Jean-Marc Thevenin	University of Toulouse 1 Capitole, France
Helmut Thoma	Thoma SW-Engineering, Basel, Switzerland
A Min Tjoa	Vienna University of Technology, Austria
Vicenc Torra	IIIA-CSIC, Spain
Traian Marius Truta	Northern Kentucky University, USA
Vassileios Tsetsos	National and Kapodistrian University of Athens, Greece
Theodoros Tzouramanis	University of the Aegean, Greece
Maria Vargas-Vera	Universidad Adolfo Ibanez, Chile
Krishnamurthy Vidyasankar	Memorial University of Newfoundland, Canada
Marco Vieira	University of Coimbra, Portugal
Jianyong Wang	Tsinghua University, China
Junhu Wang	Griffith University, Brisbane, Australia
Qing Wang	The Australian National University, Australia
Wei Wang	University of New South Wales, Sydney, Australia
Wendy Hui Wang	Stevens Institute of Technology, USA
Gerald Weber	The University of Auckland, New Zealand
Jef Wijsen	Université de Mons, Belgium

Andreas Wombacher	University Twente, The Netherlands
Lai Xu	Bournemouth University, UK
Ming Hour Yang	Chung Yuan Christian University, China (Taiwan Province)
Xiaochun Yang	Northeastern University, China
Haruo Yokota	Tokyo Institute of Technology, Japan
Zhiwen Yu	Northwestern Polytechnical University, China
Xiao-Jun Zeng	University of Manchester, UK
Zhigang Zeng	Huazhong University of Science and Technology, China
Xiuzhen (Jenny) Zhang	RMIT University, Australia
Yanchang Zhao	RDataMining.com, Australia
Yu Zheng	Microsoft Research Asia
Xiaofang Zhou	University of Queensland, Australia
Qiang Zhu	The University of Michigan, USA
Yan Zhu	Southwest Jiaotong University, China

External Reviewers

Giuseppe Amato	ISTI-CNR, Italy
Abdelkrim Amirat	University of Nantes, France
Edimilson Batista dos Santos	Federal University of Sao Joao Del Rei, Brazil
Souad Boukhadouma	University of Nantes, France
Sahin Buyrukbilen	City University of New York, USA
Changqing Chen	Yahoo, USA
Jimmy Ka Ho Chiu	La Trobe University, Australia
Camelia Constantin	UPMC (university Pierre and Marie Curie), Paris, France
Matthew Damigos	NTUA, Greece
Andrea Esuli	ISTI-CNR, Italy
Fabrizio Falchi	ISTI-CNR, Italy
Ming Fang	Georgia State University, USA
Nikolaos Fousteris	Ionian University, Greece
Maria Jesús García Godoy	Universidad de Málaga, Spain
Di Jiang	Hong Kong University of Science and Technology, Hong Kong
Christos Kalyvas	University of the Aegean, Greece
Anas Katib	University of Missouri-Kansas City, USA
Julius Köpke	University of Klagenfurt, Austria
Christian Koncilia	University of Klagenfurt, Austria
Janani Krishnamani	Georgia State University, USA
Meriem Laifa	University of Bordj Bouarreridj, Algeria
Szymon Łazaruk	Poznan University of Economics, Poland
Chien-Hsian Lee	National Sun Yat-sen University, Taiwan
Fabio Leuzzi	University of Bari, Italy
Dingcheng Li	Mayo Clinic, USA

Mr. Sheng Li	Griffith University, Australia
Lili Lin	Hohai University, China
Esteban López-Camacho	Universidad de Málaga, Spain
Dr. Luo Min	NTT, Japan
Bin Mu	City University of New York, USA
Emir Muñoz	DERI NUI Galway, Ireland
Konstantinos Nikolopoulos	City University of New York, USA
Christos Nomikos	University of Ioannina, Greece
Ermelinda Oro	ICAR-CNR, Italy
Nhat Hai Phan	LIRMM, France
Maria del Pilar Villamil	University Los Andes, Colombia
Gianvito Pio	University of Bari, Italy
Jianbin Qin	University of New South Wales, Australia
Laurence Rodrigues do Amaral	Federal University of Uberlandia, Brazil
Wei Shen	Tsinghua University, China
Sebastian Skritek	Vienna University of Technology, Austria
Vasil Slavov	University of Missouri-Kansas City, USA
Alessandro Solimando	Università di Genova, Italy
Bismita Srichandan	Georgia State University, USA
Demetris Trihinas	University of Cyprus, Cyprus
Raquel Trillo	University of Zaragoza, Spain
Dr. Yousuke Watanabe	Tokyo Institute of Technology, Japan
Alok Watve	Google, USA
Beyza Yaman	Università di Genova, Italy
Zhen Ye	University of Queensland, Australia
Shaoyi Yin	Paul Sabatier University, France
Jianhua Yin	Tsinghua University, China
Qi Yu	Rochester Institute of Technology, USA
Wei Zhang	Tsinghua University, China
Chao Zhu	The University of Michigan, Dearborn, USA

Table of Contents – Part II

AI and Databases

Matching and Searching

Information Extraction

Queries, Streams, and Uncertainty

Storage and Compression

Query Processing

Security

Distributed Data Processing

Metadata Modeling and Maintenance

Pricing and Recommending

Security and Semantics

Table of Contents – Part I

Parallel Processing

XML and RDF

Enterprise Models

Query Evaluation and Optimization

Semantic Web

Sampling

Industrial Applications

Communities

Establishing Relevance of Characteristic Features for Authorship Attribution with ANN

Urszula Stańczyk

Institute of Informatics, Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland

Abstract. Authorship attribution is perceived as a task of the paramount importance within stylometric analysis of texts. It encompasses author characterisation and comparison, and by observation and recognition of patterns in individual stylistic traits enables confirmation or rejection of authorship claims. Stylometry requires reliable textual descriptors and knowledge about their relevance for the case under study. One of the possible ways to evaluate this relevance is to employ a feature selection and reduction algorithm in the wrapper model. The paper presents research on such procedure applied to artificial neural networks used to categorise literary texts with respect to their authors, with importance of attributes discovered through sequential backward search.

Keywords: Stylometry, Authorship Attribution, Characteristic Feature, Feature Relevance, Feature Selection, Sequential Backward Search.

1 Introduction

Stylometry (or computational stylistics) is a science that investigates intricacies of writing styles. It is considered as a successor of historical textual analysis and exploits quantitative measures to express uniqueness of human thought captured in words and sentences [1]. Putting aside the subject content and categorisation of texts with respect to *what* we write about, stylometry shifts focus to *how* we do it and allows for characterisation of authors by detecting their age, gender, educational and cultural background, linguistic preferences. Through comparison of styles it extracts similarities and differentiating traits and enables recognition of authorship, which is widely perceived as the most important of its tasks [3].

Stylometry requires reliable textual descriptors, reflecting elements of writing styles. Typically there are used content-independent markers referring to function words and syntactic structures [2]. When various sets of characteristic features are proposed and not a single explanation or reasoning gives motivation for strong preference of some set over others, it is a common practice to use a set as large as possible (and feasible), and discover relevance of features by employing data mining techniques. With such approach if some features are found redundant, irrelevant, or less relevant than others, they can be discarded.

Selection and reduction of characteristic features is a widely studied problem, applicable in many domains [5]. It can be executed by *filtering*, regardless of a

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 1–8, 2013.

particular classifier employed, or the choice can be dependent on the classifier performance in the *wrapper* model. This last approach was exploited in the research presented in this paper. As a classifier used to recognise authorship for literary works artificial neural network was employed [10]. Starting with the set of arbitrarily selected characteristic features the algorithm for sequential backward search was applied next and effects of reduction for single attributes with regard to classification accuracy observed. The procedure leads to establishing an ordering of features reflecting their relevance for the case under study.

The paper is organised as follows. Section 2 presents basic stylometric notions whereas Section 3 addresses the problem of feature extraction, selection, and reduction. Section 4 provides some details of the experimental setup and Section 5 describes ANN classifier employed in the research. The process of establishing relevance of features and results of performed experiments are given in Section 6. Section 7 contains concluding remarks.

2 Stylometric Analysis

A style is an elusive phenomenon — intuitively we can tell a difference between various styles, yet how to express our subtle and individual impressions in precise terms, how to represent them in some unemotional objective measures understandable to machines and required for automatic recognition?

It is relatively easy to describe a text by its genre and topic, but as Jussi Karlgren states [1]: *Texts are much more than what they are about.* Perceiving discrepancies only in plots, characters would be extremely superficial. Even with focus on works by a single author we can observe varieties in linguistic forms and constructions. While writing, authors make choices how to capture their ideas in words and sentences, how to organise them. Some of these choices are conscious, some subconscious. When they are consistent and observable throughout all works they gave base to definitions of individual writing styles.

In order to distinguish styles there are needed textual features reflecting linguistic characteristics present and detectable in analysed samples of writing. Selection of these features is one of key issues within stylometric processing [7]. For text categorisation with respect to topic specific words or phrases are sufficient, whereas for style-based categorisation we have to go deeper and study such statistics as frequencies of usage for words, averages of word and sentence lengths, organisation of sentences into paragraphs giving syntactic structure.

To process these lexical and syntactic descriptors either statistic-oriented methodologies or techniques from artificial intelligence domain are employed [9].

3 Feature Extraction, Selection and Elimination

A task of finding characteristic features is demanding — they are typically expected to be exploited for automatic recognition, result in a high predictive accuracy, and should not be too numerous since that would complicate classification. The task takes on one of two forms:

- Feature selection — when the original form and meaning of features is preserved within processing,
- Feature extraction — when they are transformed into other measures, and we lose original semantic information.

A set of attributes describing objects for recognition could be dictated by domain knowledge, but more often we need to make a choice from the available alternatives. This selection can be executed in two approaches [5]:

- Filtering, basing on which we sieve through all features and arrive at such set that meets conditions. Filters tend to be more general and applicable in most domains as they are not dependent on any particular learning algorithm.
- Wrapping, when the choice of characteristic features is conditioned by the performance of a classifier used. Wrappers are biased by the specifics of the classifier and while obtained subsets of features can bring increase of its accuracy, they are not necessarily equally suitable for another.

In both approaches a compromise between minimality and suitability of a feature set is required as the proverbial saying of "never too much of a good thing" in case of features is not true. When there are too many features the curse of dimensionality usually works against efficiency.

It is impractical and often unfeasible to perform an exhaustive search for features, so the generation of a set of features for evaluation starts with either:

- No features, and then we add them in forward selection,
- A group of features, selected either arbitrarily or randomly, that is next expanded or reduced,
- All features, some of which are later rejected in backward elimination.

Observations on how elimination of features affects the classifier performance bring information about relevance of the studied attributes for the problem to be solved, as illustrated in this paper.

4 Input Datasets

The first rule of stylometric analysis, or, in fact, for any pattern recognition and classification task, is that we need access to some sufficiently large number of samples, *sufficient* being the operative word in this statement. Basing on these samples the characteristic features are next selected or extracted. Once they are acquired, a data processing technique can be employed and a classifier constructed. Its performance depends on many aspects, whether it is suitable for the task, if descriptive properties of features express characteristics of data samples, whether there are enough samples, if they are representative.

In the research on authorship recognition as the corpus there were used selected literary works by Jane Austen and Edith Wharton, available thanks to Project Gutenberg (`http://www.gutenberg.org`), as listed in Table 1. Since in such long novels as these some variations of styles can be observed, they were

Table 1. Literary works analysed within stylometric research

	Jane Austen	Edith Wharton
Learning dataset	"Emma"	"A Backward Glance"
	"Mansfield Park"	"The Age of Innocence"
	"Pride and Prejudice"	"The Glimpse of the Moon"
	"Persuasion"	"The Reef"
Testing dataset	"Northanger Abbey"	"Certain People"
	"Sense and Sensibility"	"House of Mirth"
	"Lady Susan"	"Summer"

divided into smaller samples of writing, of comparable length, typically corresponding to chapters or their parts, few pages or about 4 thousands words long, the same number of samples for both authors in both datasets.

To describe samples we need textual markers, referring to lexical and syntactic elements of a linguistic style. Therefore, the frequencies of usage were considered for punctuation marks and some function words selected basing on the list of the most popular words in English language, as follows:

- Syntactic descriptors reflected the usage of a comma, a colon, a semicolon, a bracket, a hyphen, a fullstop, a question mark, an exclamation mark.
- Lexical markers: but, and, not, in, with, on, at, of, as, this, that, by, for, to, if, what, from.

Together both types of descriptors amount to 25 attributes, the set employed in the past research works dedicated to authorship attribution [8]. For this set of features the base connectionist classifier was next built.

5 ANN Classifier

Connectionist classifiers are an alternative to rule-based solutions in cases where observation of subtle patterns in data is needed. Originally invented to model the work of biological neurons and their systems [4], artificial neural networks infer knowledge from the available data samples by adapting their internal structure.

Multilayer Perceptron (MLP), employed in the presented research, is a feed-forward network constructed from an input, output, and some hidden interconnected layers, with the number of neurons in them determined by the specifics of a classification task. Typically MLP employs supervised learning by backpropagation. In this algorithm the network adjusts a vector of weights associated with connections ($\mathbf{W}$) in order to minimise the error on the output, equal to the difference between the expected d_i^m and the obtained outcome $y_i^m(\mathbf{W})$, for all I output neurons and for all M learning facts:

$$e(\mathbf{W}) = \frac{1}{2} \sum_{m=1}^{M} \sum_{i=1}^{I} (d_i^m - y_i^m(\mathbf{W}))^2 \tag{1}$$

In the experiments performed California Scientific Brainmaker simulation software was exploited. To define a network, an operator needs to choose an activation function for neurons (sigmoid was selected), which determines when they fire, and the number of neurons and layers. Two outputs corresponded to two recognised classes (authors), while 25 inputs were dedicated to the original selection of textual attributes. The number of hidden layers and neurons is an important parameter of ANN performance, often established in tests, the results of which are given in Table 2. To minimise the effect of the initiation of weights executed at the beginning of each learning phase, multi-starting procedure was used and each network was trained 20 times. The results are listed in three categories: the worst, the best, and median classification accuracy.

Table 2. Performance of ANN classifier in relation to the network structure

Number of hidden layers	0	1			2				
Number of neurons					25	25	14	21	19
in hidden layers	0	25	14	7	25	2	13	6	6
	Classification accuracy [%]								
Minimal	90.00	90.00	90.00	90.00	90.00	90.00	88.89	90.00	90.00
Median	90.00	91.11	91.11	91.11	91.11	91.11	91.11	91.11	91.11
Maximal	91.11	91.11	92.22	92.22	93.33	92.22	91.11	92.22	93.33

All tested ANNs returned acceptable results, yet in order as not to worry about linear separability of the problem to be solved, we need a network with two hidden layers, and from these constructed the best results for the fewest neurons are given in the right-most column of Table 2. The total number of hidden neurons equals that of inputs, and they are divided into two layers: to the second the integer part of one fourth, the rest to the first layer. This rule was employed for all networks tested within the presented research.

6 Relevance of Characteristic Features

In this age of optimisation, construction of classifiers which return satisfactory results is rarely considered as the end of the road. Even if attempts at increasing the performance fail, it is informative to discover the relevance of features for the considered classification task and expose their individual influence.

The wrapper approach to feature selection and reduction requires the processing technique used not to possess its own mechanisms dedicated to dimensionality reduction. It is not entirely true for artificial neural networks since there exist several pruning algorithms [6], which enable detection and removal of such neurons that by low weights of interconnections have relatively little influence on the output. However, these algorithms are not inherent parts of a network.

In the research to establish relevance of characteristic features in authorship attribution task, the sequential backward elimination procedure was exploited. Starting with the original set of 25 attributes, the classifier performance was tested when one feature was rejected. From all alternatives the one with the best predictive accuracy was selected as the starting point for the next stage of execution, where the process was repeated and another attribute discarded. The procedure stops when there are no features left to reduce or some other conditions met. For N studied features there are up to $N-1$ stages, and up to

$$N + (N-1) + (N-2) + \cdots + 3 + 2 = \frac{(N+2)(N-1)}{2} \tag{2}$$

networks to be tested, which for high N can become prohibitive. Also, the elimination procedure imposes an ordering of features within which the search is not exhaustive. Once a feature is rejected it is not reconsidered again. Despite these drawbacks application of the described methodology can result in increased predictive accuracy for some reduced number of inputs, it also provides information about redundancy of some features while indicating those of higher relevance, which can be compared against other classifiers and other tasks.

The results of conducted experiments are given in Table 3 and Fig. 1. Table 3 specifies the list of remaining features and the one eliminated at each stage of execution, along with the classification accuracy for each best network. It also indicates ranking of features corresponding to their relevance.

Fig. 1 displays these results in graphs, plotting the minimal, median and best predictive accuracies for all networks. It is apparent, that with this procedure even when reduction of features reaches 84% (21 out of 25 eliminated and only 4 left), the resulting artificial neural network classifies with better accuracy than the base network with the complete set of 25 attributes.

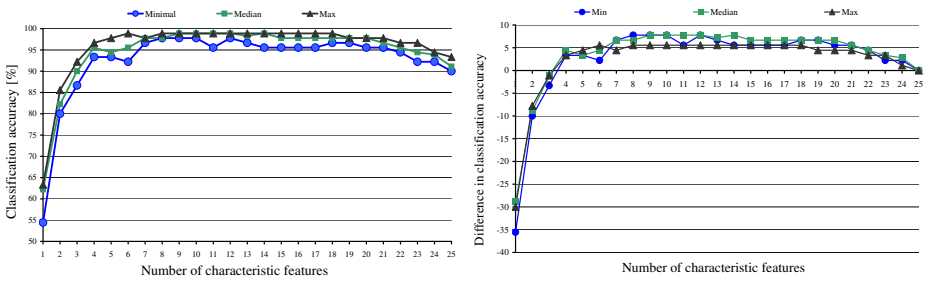


Fig. 1. Classification accuracy in relation to the number of features employed, expressed as: a) an absolute value, b) a calculated difference (increase + or decrease -) with respect to the base performance for all characteristic features

The relatively simple procedure described enables to detect these characteristic features which can be treated as either irrelevant or redundant for the particular classification task, and, when discarded, can bring significant decrease in

Table 3. Establishing the relevance of characteristic features by sequential backward elimination. The stage of execution with the index 0 is the starting point where all features are involved in classification. In subsequent stages with increasing indices one feature at a time is eliminated based on the classifier performance.

Elimination stage	Features remaining	Eliminated feature	Classification accuracy [%]
0	but and not in with on at of as this that by for to if what from . , ; : ! ? (-		91.11
1	but and not in with on at of as this that by for to if what from . ; : ! ? (-	,	93.89
2	but and not in with on at of as this that by for to if what from . ; : ! ? -	(	94.44
3	but and not in with on at of as this that by for to if what from . ; : ! ?	-	95.56
4	but and not in with on of as this that by for to if what from . ; : ! ?	at	96.67
5	but and not in on of as this that by for to if what from . ; : ! ?	with	97.78
6	but and not in on of as this that by for to if from . ; : ! ?	what	97.78
7	but and not in on of as this that by for to if . ; : ! ?	from	97.78
8	but and not in on of as this that by for if . ; : ! ?	to	97.78
9	but and not in on of as this that by if . ; : ! ?	for	97.78
10	but and not in on as this that by if . ; : ! ?	of	97.78
11	but and not in on as this that by if ; : ! ?	.	98.89
12	but and not on as this that by if ; : ! ?	in	98.33
13	but and not on as this that by if ; : ?	!	98.89
14	but and not on as that by if ; : ?	this	98.89
15	and not on as that by if ; : ?	but	98.89
16	and not on as by if ; : ?	that	98.89
17	and not on as by ; : ?	if	97.78
18	and not on as by ; :	?	97.78
19	not on as by ; :	and	95.56
20	not on as ; :	by	94.44
21	not on as ;	:	95.56
22	not on ;	as	90.00
23	not ;	on	82.22
24	not	;	62.22

the classifier size and even noticeable increase of its performance. On the other hand, the lack of generality due to wrapper approach and additional computational and processing costs involved can be considered as certain disadvantages.

7 Conclusions

The paper presents research on authorship attribution performed with ANN classifier. Starting with the set of arbitrarily selected characteristic features, the sequential backward elimination procedure is employed, where at each stage one attribute is reduced. From all alternatives there is selected one with the highest predictive accuracy. The process stops when there are no features left to reject. With this approach not only we can discover the network which performs better for fewer inputs but also expose the influence of individual attributes on classification accuracy reflecting their relevance. Since in the search there is considered a part of possible feature subsets the methodology cannot guarantee finding the best solution, and for a high number of features the additional computational costs involved can become prohibitive, yet the process is relatively simple and it gives a reasonable chance of arriving at such subset of features for which the performance is improved and the structure of the classifier reduced. In the future research the procedure will be employed and tested for rule-based classifiers.

References

1. Argamon, S., Burns, K., Dubnov, S. (eds.): The structure of style: Algorithmic approaches to understanding manner and meaning. Springer, Berlin (2010)
2. Burrows, J.: Textual analysis. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) A Companion to Digital Humanities. Blackwell, Oxford (2004)
3. Craig, H.: Stylistic analysis and authorship studies. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) A Companion to Digital Humanities. Blackwell, Oxford (2004)
4. Fiesler, E., Beale, R.: Handbook of neural computation. Oxford University Press (1997)
5. Jensen, R., Shen, Q.: Computational Intelligence and Feature Selection. John Wiley & Sons, Inc., Hoboken (2008)
6. Kavzoglu, T., Mather, P.: Assessing artificial neural network pruning algorithms. In: Proceedings of the 24th Annual Conference and Exhibition of the Remote Sensing Society, Greenwich, UK, pp. 603–609 (2011)
7. Peng, R., Hengartner, H.: Quantitative analysis of literary styles. The American Statistician 56(3), 15–38 (2002)
8. Stańczyk, U.: Rough set-based analysis of characteristic features for ANN classifier. In: Graña Romay, M., Corchado, E., Garcia Sebastian, M.T. (eds.) HAIS 2010, Part I. LNCS, vol. 6076, pp. 565–572. Springer, Heidelberg (2010)
9. Stańczyk, U.: Rough set and artificial neural network approach to computational stylistics. In: Ramanna, S., Howlett, R., Jain, L. (eds.) Emerging Paradigms in Machine Learning. SIST, vol. 13, pp. 441–470. Springer, Heidelberg (2013)
10. Waugh, S., Adams, A., Tweedie, F.: Computational stylistics using artificial neural networks. Literary and Linguistic Computing 15(2), 187–198 (2000)

Genetic Programming with Greedy Search for Web Service Composition

Anqi Wang, Hui Ma, and Mengjie Zhang

Victoria University of Wellington, New Zealand
andy.wanganqi@gmail.com, {hui.ma,mengjie.zhang}@ecs.vuw.ac.nz

Abstract. Service compositions build new web services by orchestrating sets of existing web services provided in service repositories. Due to the increasing number of available web services, the search space for finding best service compositions is growing exponentially. In this paper, a combination of genetic programming and random greedy search is proposed for service composition. The greedy algorithm is utilized to generate valid and locally optimized individuals to populate the initial generation for genetic programming, and to perform mutation operations during genetic programming. A full experimental evaluation has been carried out using public benchmark test cases with repositories of up to 15,000 web services and 31,000 properties. The results show good performance in searching for best service compositions, where the number of atomic web services used and the tree depth are used as objectives for minimization.

1 Introduction

Service-oriented software is built on top of service repositories containing hundreds or thousands of atomic web services. Due to the increasing number of available services, the search space for finding the best service composition is growing exponentially. Hence, computing optimal solutions is impractical in general. Rather, one is interested in efficient and effective approaches for computing near-optimal solutions.

Web service composition has recently attracted much interest. Many existing approaches tackle service composition tasks by considering them as planning problems using established planning techniques [5, 8, 13–16]. However, these approaches do not scale. The complexity that they consider is much lower than the one typically observed in service composition tasks based on dedicated web service languages like OWL-S [12] and BPEL4WS [1]. Other approaches tackle service composition tasks by using artificial intelligence techniques [18, 20, 21]. Most approaches have been tested for small service repositories only, without any attention to scalability. In [2, 17, 19], genetic programming (GP) is used for computing near-optimal service compositions. A thorough analysis reveals the limited effectiveness of the evolutionary process in these GP-based approaches that is due to the complexity of the data structures used and the randomness of the initial population. Therefore, it requires extremely long time to discover near-optimal solutions, and the results are very unstable, see [17].

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 9–17, 2013.

The goal of this paper is to propose a novel GP-based approach to web service composition that overcomes shortcomings of previous GP-based approaches. Instead of starting with an initial population of service compositions that are randomly generated from the huge number of atomic web services in the repository, we apply a greedy search algorithm to pre-filter the repository for those atomic web services that are exclusively related to the given service composition task. We have examined our proposal using the public web service repositories of OWL-S TC [11], WSC2008 [4] and WSC2009 [9] as benchmarks. Specifically, we have investigated the following objectives:

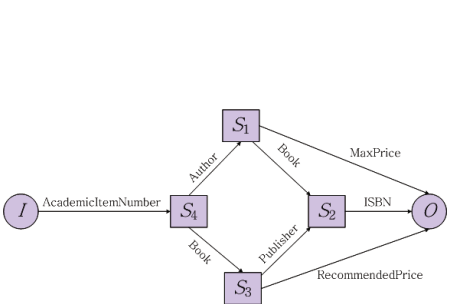
1. Whether the new method can achieve reasonably good performance, and in particular outperforms existing GP-based approaches.
2. Whether the greedy algorithm can effectively discover atomic web services that are exclusively related to the service composition task.
3. Whether the evolved program (the solution to the given service composition task) is interpretable.

This paper is structured as follows: Section 2 discusses representations of service compositions. Sections 3 and 4 present our approach, while Section 5 reports on the experiments conducted to test it. Finally, Section 6 states our conclusions.

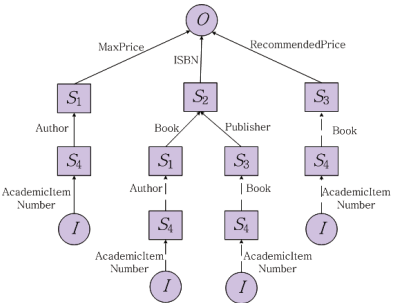
2 Representation of Service Compositions

Web services for complex tasks can be composed from atomic web services provided in the service repository. A web service takes certain inputs to generate certain outputs. The inputs and outputs can be semantically described through ontologies as *concepts*, cf. [4, 9]. Assume, for example, the given task is to determine the maximum price of a book, its ISBN and the recommended price in dollars for a given input AcademicItemNumber. That is, a web service is needed that takes the concept $I = \{AcademicItemNumber\}$ as input, and produces the concept $O = \{MaxPrice, ISBN, RecommendedPrice\}$ as output, see Fig. 1. If the service repository contains no such atomic web service, a composite service might be able to accomplish the given task.

Service compositions are often represented as directed acyclic graphs, see Fig. 1(a). Squares represent atomic web services used in the composition, while circles represent the input concept I and the output concept O of the composite service S. Arcs are labelled by the properties that are transferred from one atomic service to another, or taken from the input I, or produced for the output O. The service composition must be verified for formal correctness. For that, each atomic web service used in the composition must satisfy the *matching rule*: its input concept must be subsumed by the union of the properties on its incoming arcs, and its output concept must subsume the union of the properties on its outgoing arcs. In this case, the input concept *matches* the properties received, and the output concept *matches* the properties sent. In our example, all atomic web services satisfy their respective matching rules.



(a) Graph representation of S



(b) Tree representation of S

	Atomic web service	Input	Output
S_1	GetMaxPrice	Author	Book, MaxPrice
S_2	GetISBN	Book, Publisher	Author, ISBN
S_3	GetRecommendedPricePublisher	Book	RecommendedPrice, Publisher
S_4	GetBookAuthor	AcademicItemNumber	Book, Author

(c) Some atomic web services in the repository

Fig. 1. A composite web service S composed from four atomic web services $S_1, \ldots, S_4$ found in the repository

The number of candidate service compositions grows exponentially with the number of atomic web services in the repository and with the number of properties involved. Due to the inherent complexity of the service composition problem we employ GP to find near-optimal solutions. We assume that the reader is familiar with the principles of *genetic programming* (GP) [10]. While directed acyclic graphs are a natural way to represent service compositions, GP traditionally represents evolved programs as tree structures in memory.

To facilitate the use of tree-based GP techniques the graph representation of service compositions is converted into a tree representation. We make use of the standard transformation of directed acyclic graphs into trees [3], also known as *unfolding*. In our example, the directed acyclic graph in Fig. 1(a) is transformed into the tree in Fig. 1(b). Unfolding starts with the output concept O which becomes the root of the tree, while the terminal nodes of the tree represent multiple copies of the input concept I of the composite service S. Unfolding often causes duplicate nodes. In Fig. 1(b), there are two S_1 nodes, two S_3 nodes, four S_4 nodes, and four I nodes. For the sake of simplicity, we occasionally skip the subtree rooted at a duplicated node in our illustrations.

Related Work. Several GP-based approaches for web service composition have been proposed. [2] pioneered the use of GP by introducing genetic operators and fitness functions for service compositions, but tested only with very small repositories. [17] proposed a tree representation of individuals, with internal nodes for control structures and terminal nodes for atomic web services, and used a context-free grammar to randomly initialize the first generation for GP. By construction, the initial generation contains many weak individuals, thus making

the approach inefficient and unstable. To overcome shortcomings of [17], [19] refined the tree representation, and introduced the service dependency graph for checking the matching rules. A major limitation is that only atomic web services with a single property as output are permitted. Other approaches work with graph representations of service compositions [20, 21], but are not GP-based. [7], for example, used breath first search to compute valid service compositions, but made no efforts to find best solutions.

3 The Novel GP-Based Approach

Now, we define the variables commonly used in GP, i.e., the terminal set, the function set, and the fitness function [10]. A *service composition task* is defined by an input concept I, an output concept O, and a repository R of atomic web services. We use the atomic web services in the given repository as the *function set* in GP, i.e., we regard the atomic web services as *functions* that map inputs to outputs. GP uses the tree representation discussed above: the internal nodes correspond to functions, all terminal nodes to the input concept I, and the root to the output concept O.

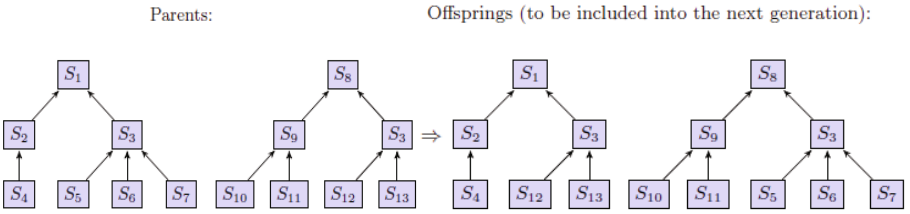


Fig. 2. An example for our crossover

A *fitness function* is used to measure the quality of candidate compositions. We use the unduplicated number of atomic web services used in a service composition to measure the fitness of a service composition. The fewer atomic web services used in the service composition, the better its performance will be. Occasionally, we also use the tree depth to measure the fitness of service compositions. The tree depth corresponds to the length of the longest path from the input concept to the output concept. In our example in Fig. 1, the number feature is 4, since $S_1, \ldots, S_4$ are used and duplicates are not considered, and the depth feature is 3. We use the depth feature only to distinguish service compositions with identical number feature. If two service compositions share their number feature, shallow trees are preferred.

GP uses the operations crossover, mutation, and reproduction to evolve individuals, i.e., service compositions in our case. To perform *crossover*, we stochastically select two random individuals and check if there is one node representing

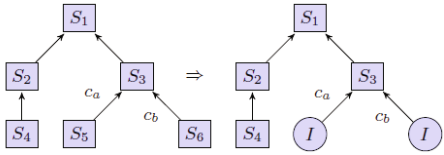


Fig. 3. An example for our mutation

the same atomic web service in both individuals, and then swap the node together with their subtrees between the two individuals. This guarantees that the matching rules stay satisfied. In Fig. 2, for example, the S_3 nodes in the two individuals are swapped together with the subtrees rooted at them. As usual, the two new individuals generated as offsprings from the two individuals from the previous generation are then included into the next generation.

The *mutation* operator is normally used to replace a node together with its subtree in a selected individual, or to replace only the node. In our approach, we perform mutation by stochastically selecting one node in a randomly chosen individual and replacing its subtree with a new subtree generated by applying a greedy algorithm that will be presented in Section 4. In Fig. 3, for example, assume S_3 is selected for mutation and c_a and c_b are properties that S_3 receives from S_5 and S_6, respectively. Then, the mutation operator replaces the subtree of S_3 with a new subtree to generate a new individual as an offspring.

The fitness of the offspring generated by crossover or mutation can be smaller than its parents' one. To avoid a decrease of fitness of the fittest individuals we choose a top percentage of individuals from the old generation for mere *reproduction* and include them into the next generation without any modification.

4 Random Greedy Search for Initialization and Mutation

Next we propose a random greedy algorithm for computing locally optimal solutions for a service composition problem, see Algorithm 1. Its inputs are the input concept I, the output concept O, and the repository R of the service composition task to be solved. The algorithm generates the tree representation of a service composition S that is formally correct for the composition task at hand.

In the algorithm, C_{search} denotes the concept used for searching the repository R, and S_{found} denotes the set of all those atomic web services whose inputs match C_{search}. To begin with, C_{search} is initialized by the input concept I. The discovered atomic web services are added to S_{found}, the outputs of these services are adjoined to C_{search}. Steps 4 to 8 are repeated until no new atomic web service is discovered. This is in particular the case when C_{search} is no longer extended. Afterwards it is checked whether C_{search} subsumes the required output concept O of the composition task. If so, then the composition task has a solution. By applying the matching rule, the nodes of the tree are then stochastically connected to generate the arcs of the tree. Otherwise, there is no solution.

Input: I, O, R
Output: a service composition S
1: $C_{search} \leftarrow I$;
2: $S_{list} \leftarrow \{\}$;
3: $S_{found} \leftarrow DiscoverService()$;
4: **while** $|S_{found}| > 0$ **do**
5: $S_{list} \leftarrow S_{list} \cup S_{found}$;
6: $C_{search} \leftarrow C_{search} \cup C_{output}$ of S_{found};
7: $S_{found} \leftarrow DiscoverService()$;
8: **end while**
9: **if** $C_{search} \supseteq O$ **then**
10: $ConnectNodes()$;
11: Report solution;
12: **else**
13: Report no solution;
14: **end if**

Algorithm 1. A greedy algorithm for service composition.

We use the random greedy algorithm as an auxiliary to our GP-based approach in Section 3. It generates a set of locally optimal individuals to populate the initial generation for GP. By construction, all of them are formally correct solutions for the composition task at hand. By using the search concept C_{search} the algorithm only considers services that are related to the composition task. The locally optimal individuals constitute an initial population that is already of high quality, thus overcoming a weakness of previous GP-based approaches.

Moreover, we apply our greedy algorithm to perform mutation in our GP-based approach. We use it to generate the new subtree rooted at the selected node. This time, the output of the corresponding atomic web service serves as O, and R is restricted to the atomic web services that occur in the composition.

5 Empirical Results

To evaluate our proposal we tested it using the collection of benchmark test cases provided by the web service competitions WSC2008 and WSC2009. Each test case specifies a service composition task including input concept, output concept, and service repository. The complexity of the composition tasks is very diverse in terms of the overall number of properties considered. In addition, we also used the benchmark test cases of OWL-S TC V2.2 for testing. While the repositories of WSC2008 and WSC2009 are all randomly generated, the repositories of OWL-S TC are all from real domains.

Our test platform was a PC with an i5-3320(2.60GHz) processor, 4.0 GB RAM, and Windows 7 64-bit operating system. As our approach is stochastic, we run each task 30 independent times to record the average and standard deviation of the best fitness and the time consumed. Clearly, the population size and the number of generations used for GP affected the time. For our tests we set

Table 1. Original repositories vs. shrunk repositories from greedy search

Task	|Atomic Web Services|		|Properties|	
	Original	Shrunk	Original	Shrunk
WSC2008-1	158	61	1540	252
WSC2008-2	558	63	1565	245
WSC2008-3	604	106	3089	406
WSC2008-4	1041	45	3135	205
WSC2008-5	1090	103	3067	423
WSC2008-6	2198	205	12468	830
WSC2008-7	4113	165	3075	621
WSC2008-8	8119	132	12337	596
WSC2009-1	572	80	1578	331
WSC2009-2	4129	140	12388	599
WSC2009-3	8138	153	18573	644
WSC2009-4	8301	330	18673	1432
WSC2009-5	15211	237	31044	1025

the parameters population size = 200, number of generations = 30, reproduction percentage = 0.1, crossover percentage = 0.8, mutation percentage = 0.1, and left the tree depth unbounded.

To evaluate the effectiveness of our greedy algorithm we applied it to all test cases provided by WSC2008 and WSC2009. Table 1 shows the number of atomic web services and the number of properties used for the original repositories, and for the repositories shrunk by applying the greedy algorithm for initialization. For example, for task 1 of WSC2008, there are 158 atomic web services in the original repository, but only 61 of them are related to the given composition task. This demonstrates the effectiveness of our greedy algorithm for reducing the search space.

Table 2 shows a comparison of our approach with a recent approach proposed in [17] that also used OWL-S TC, WSC2008, and WSC2009 for testing. Column "min" records the number of atomic web services in the best known solutions, see [4, 9, 11]. There are three columns for our approach: Column "number" records the number of atomic web services in the best solution found by our approach, column "depth" records the tree depth of the best solution, and column "time(ms)" records the search time used for computing the best solution. For the existing approach [17] the respective information is given in the remaining three columns. Note that the search times recorded for the two approaches are not directly comparable as they were evaluated on different platforms.

Our approach was successful in computing a solution for each of the service composition tasks specified by WSC2008 and WSC2009, except for tasks 9 and 10 of WSC2008 which are both known not to have a solution [4]. Recall that our approach only needs the initial greedy search to check for the mere existence of a solution, and is therefore very efficient.

Note that [17] tested their approach with only 5 tasks from OWL-S TC and 3 tasks from WSC2008. Our approach achieved good test results for the remaining tasks, too. For all solvable tasks, our solutions are interpretable and the numbers of services in our solutions are equal or very close to the numbers of services in the best known solution, with less than 1.00 of standard deviation. In terms of

Table 2. Average results for the tests ($\bar{\chi} \pm \sigma$)

Task		Our Approach			Existing Approach [17]		
name	min	number	depth	time(ms)	number	depth	time(ms)
OWL-S TC1	1	1.00 ± 0.00	1.00 ± 0.00	14 ± 10	1.00 ± 0.00	1.00 ± 0.00	749 ± 364
OWL-S TC2	2	2.00 ± 0.00	2.00 ± 0.00	51 ± 15	2.00 ± 0.00	2.00 ± 0.00	484 ± 139
OWL-S TC3	2	2.00 ± 0.00	2.00 ± 0.00	250 ± 16	2.00 ± 0.00	2.00 ± 0.00	473 ± 76
OWL-S TC4	4	4.00 ± 0.00	2.63 ± 0.49	341 ± 15	5.70 ± 1.19	2.20 ± 0.40	3010 ± 422
OWL-S TC5	3	3.00 ± 0.00	1.00 ± 0.00	389 ± 21	3.30 ± 0.46	1.00 ± 0.00	1098 ± 240
WSC2008-1	10	10.00 ± 0.00	3.00 ± 0.00	2338 ± 33	15.80 ± 5.71	6.00 ± 1.26	6919 ± 1612
WSC2008-2	5	5.00 ± 0.00	3.87 ± 0.35	2172 ± 23	6.00 ± 0.89	3.50 ± 0.67	11137 ± 3106
WSC2008-3	40	40.60 ± 0.62	23.00 ± 0.00	145135 ± 2102	n/a	n/a	n/a
WSC2008-4	10	10.00 ± 0.00	5.00 ± 0.00	1301 ± 31	n/a	n/a	n/a
WSC2008-5	20	20.00 ± 0.00	8.00 ± 0.00	7630 ± 52	49.90 ± 16.84	9.20 ± 2.96	95390 ± 43521
WSC2008-6	40	45.80 ± 0.92	9.00 ± 0.00	38935 ± 297	n/a	n/a	n/a
WSC2008-7	20	20.00 ± 0.00	15.00 ± 0.00	25433 ± 385	n/a	n/a	n/a
WSC2008-8	30	32.10 ± 0.30	23.00 ± 0.00	27123 ± 257	n/a	n/a	n/a
WSC2008-9	n/a	n/a	n/a	n/a	n/a	n/a	n/a
WSC2008-10	n/a	n/a	n/a	n/a	n/a	n/a	n/a
WSC2009-1	5	5.00 ± 0.00	3.67 ± 0.96	4986 ± 29	n/a	n/a	n/a
WSC2009-2	20	20.03 ± 0.18	6.00 ± 0.00	19086 ± 93	n/a	n/a	n/a
WSC2009-3	10	10.20 ± 0.76	3.07 ± 0.25	19173 ± 179	n/a	n/a	n/a
WSC2009-4	40	42.03 ± 0.85	8.00 ± 4.32	192930 ± 3288	n/a	n/a	n/a
WSC2009-5	30	30.07 ± 0.25	19.00 ± 0.00	93209 ± 382	n/a	n/a	n/a

search time, our results are also stable with standard deviation less than 5% of the average. The time consumed is short, even for the most complex tasks of WSC2008 (task 3) and WSC2009 (task 4).

The evaluation results show that our proposed web service composition approach can efficiently produce correct and interpretable near-optimal solutions.

6 Conclusions

In this paper we presented an approach for performing web service composition using a combination of GP and greedy search. The random greedy algorithm is an auxiliary to GP. It generates locally optimal individuals for populating the initial generation for GP, and to perform mutations during GP. Moreover, it guarantees that the generated individuals are formally correct and thus interpretable for web service composition. We have tested our approach with service composition tasks from the common benchmark test case collections. The analysis of the experimental results shows that our approach is efficient, effective and very stable for computing near-optimal solutions. Most notably, the initial greedy search helps to shrink the number of atomic web services to be considered by GP later on, thus greatly reducing the search space.

References

1. Andrews, T.: Business Process Execution Language for Web Services (2003)
2. Aversano, L., di Penta, M., Taneja, K.: A genetic programming approach to support the design of service compositions (2006)
3. Bang-Jensen, J.: Digraphs: Theory, Algorithms and Applications. Springer (2008)

4. Bansal, A., Blake, M., Kona, S., Bleul, S., Weise, T., Jaeger, M.: WSC-2008: Continuing the web services challenge. In: IEEE Conf. on E-Commerce Technology, pp. 351–354 (2008)
5. Carman, M., Serafini, L., Traverso, P.: Web service composition as planning. In: ICAPS Workshop on Planning for Web Services (2003)
6. Elmaghraoui, H., Zaoui, I., Chiadmi, D., Benhlima, L.: Graph-based e-government web service composition. CoRR, abs/1111.6401 (2011)
7. Hashemian, S., Mavaddat, F.: A graph-based approach to web services composition. In: Symposium on Applications and the Internet, pp. 183–189 (2005)
8. Klusch, M., Gerber, A.: Semantic web service composition planning with OWLS-XPlan. In: Int. AAAI Symposium on Agents and the Semantic Web (2005)
9. Kona, S., et al.: WSC-2009: A quality of service-oriented web services challenge. In: IEEE Int. Conf. on Commerce and Enterprise Computing, pp. 487–490 (2009)
10. Koza, J.: Genetic Programming. MIT Press (1992)
11. Kuster, U., Konig-Ries, B., Krug, A.: An online portal to collect and share SWS descriptions. In: IEEE Int. Conf. on Semantic Computing, pp. 480–481 (2008)
12. Martin, D., et al.: OWL-S Semantic Markup for Web Services (2004)
13. Oh, S.-C., Lee, D., Kumara, S.: Effective web service composition in diverse and large-scale service networks. IEEE Trans. Services Comp. 1(1), 15–32 (2008)
14. Oh, S.-C., Lee, D., Kumara, S.R.T.: A comparative illustration of AI planning-based web services composition. SIGecom Exch. 5(5), 1–10 (2006)
15. Pistore, M., Marconi, A., Bertoli, P., Traverso, P.: Automated composition of web services by planning at the knowledge level. In: IJCAI, pp. 1252–1259 (2005)
16. Rao, J., Küngas, P., Matskin, M.: Composition of semantic web services using linear logic theorem proving. Inf. Syst. 31(4), 340–360 (2006)
17. Rodriguez-Mier, P., Mucientes, M., Lama, M., Couto, M.: Composition of web services through genetic programming. Evolutionary Intelligence 3, 171–186 (2010)
18. Xia, H., Chen, Y., Li, Z., Gao, H., Chen, Y.: Web service selection algorithm based on particle swarm optimization. In: IEEE DASC, pp. 467–472 (2009)
19. Xiao, L., Chang, C., Yang, H.I., Lu, K.S., Jiang, H.Y.: Automated web service composition using genetic programming. In: IEEE COMPSAC, pp. 7–12 (2012)
20. Yang, Z., Shang, C., Liu, Q., Zhao, C.: A dynamic web services composition algorithm. J. Comp. Inform. Syst. 6(8), 2617–2622 (2010)
21. Zhang, W., Chang, C.K., Feng, T., Jiang, H.Y.: QoS-based dynamic web service composition with ant colony optimization. In: IEEE COMPSAC, pp. 493–502 (2010)

Multivariate Prediction Based on the Gamma Classifier: A Data Mining Application to Petroleum Engineering

Itzamá López-Yáñez[1,2], Leonid Sheremetov[1], and Oscar Camacho-Nieto[2]

[1] Mexican Petroleum Institute (IMP),
Av. Eje Central Lázaro Cárdenas Norte, 152, Mexico City, Mexico
{itzamal,sher}@imp.mx
[2] Instituto Politécnico Nacional – Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC - IPN),
Av. Juan de Dios Bátiz s/n Edificio CIDETEC, Mexico City, Mexico
oscarc@cic.ipn.mx

Abstract. A novel associative model was developed to predict petroleum well performance after remedial treatments. This application is of interest, particularly for non-uniform oilfields such as naturally fractured ones, and can be used in decision support systems for water control or candidate well selection. The model is based on the Gamma classifier, a supervised pattern recognition model for mining patterns in data sets. The model works with multivariate inputs and outputs under the lack of available data and low-quality information sources. An experimental dataset was built based on historical data of a Mexican naturally fractured oilfield. As experimental results show, this classifier-based predictor shows competitive performance compared against other methods.

Keywords: data mining, Gamma classifier, prediction, petroleum engineering.

1 Introduction

Data mining (DM) techniques permit exploring large amounts of data in search of consistent patterns and/or interesting relationships between variables [1]. They provide tools for data analysis enabling better decision making in different domains. In Earth sciences, pattern recognition techniques have shown to be very useful for solving such DM tasks as segmentation, classification, and indexing [2]; recently they were used for prediction [3]. Nevertheless, despite such advantages of associative models as low complexity, high capacity for non-linearity, or high performance, their feasibility as a tool for multivariate prediction has not been fully explored yet.

In previous papers the Gamma classifier was introduced, which is a supervised pattern classifier of recent emergence, belonging to the associative approach to pattern recognition. Recently this model has been extended to univariate time series forecasting, for one-step-ahead and long-term scenarios, where it showed competitive results [4, 5]. Here, said model is extended to multivariate mixed prediction, where some variables are time series fragments, others are categorical and yet others are numerical.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 18–25, 2013.

The contribution of the present work is twofold. First, an algorithm for multivariate prediction based on the Gamma classifier is developed; then, tests for multivariate prediction were performed on a specific data set of oil wells workovers. The main feature of the case study is a small data sample, containing only 43 instances. The experimental results show that for these data sets, the proposed model can be preferable in terms of prediction accuracy compared to well-known techniques, such as Bayesian network and the *k*-Nearest Neighbor (*k*-NN) algorithm [6, 7].

The rest of the paper is organized as follows. In the next section the motivation for research is discussed. Section 3 presents the Gamma classifier, which is the basis for the proposal. The methodology for developing the prediction model is further discussed in section 4. Section 5 presents the experimental results and discusses the advantages and limitations of each model, followed by concluding remarks.

2 Motivation for Research

During oil and gas production nearly all wells go through remedial treatments involving invasive techniques to improve productivity. These operations are called workovers and interventions, and may include stimulation treatments like hydraulic fracturing, matrix treatments, gel treatment, or replacement of damaged production tubing. During interventions production may be suspended for several days. One of the most important steps for successful interventions is to identify candidate wells.

Traditionally, in the oil and gas industry the selection of wells to be treated has been based on complicated reservoir models or, often, unreliable anecdotal screening guidelines, which may result in inconsistent outcomes and poor performance [8]. Recently machine learning methods have been used to analyze pre- and post-treatment historical data for gel-polymer water treatments [9, 10], showing a dramatic improvement in success rates. Neural networks (ANN) prediction was applied also to estimate monthly oil production potential for infill wells [11]. The ANN provided predicted production for each gridpoint of the oilfield, about 64K points, giving a correlation coefficient of 0.79.

The main requirement for their application is that enough data are available to train the networks, which is not always the case. Thus, here we propose a different prediction method based on the Gamma classifier. Though the proposal can be considered as a generic one for remedial treatment applications, a particular case of matrix treatment was selected. The prediction accuracy is tested and the relative importance of various predictors is evaluated.

3 Gamma Classifier

This supervised learning associative model was originally designed for pattern classification, and borrows its name from the operation at its heart: the Gamma operator [4, 5]. This operation takes as input two binary patterns — $\mathbf{x}$ and $\mathbf{y}$ — and a non-negative integer θ and gives a 1 if both vectors are similar or 0 otherwise. Other operations (α, β, and u_β) used by the Gamma operator are also included.

Definition 1 (Alpha and Beta operators). Given the sets $A = \{0,1\}$ and $B = \{0,1,2\}$, the alpha (α) and beta (β) operators are defined in a tabular form as shown in Table 1. Their corresponding vector versions for inputs $\mathbf{x} \in A^n$ and $\mathbf{y} \in A^n$ give an n-dimensional vector as output, whose i-th component is computed as follows.

$$\alpha(\mathbf{x},\mathbf{y})_i = \alpha(\mathbf{x}_i,\mathbf{y}_i) \text{ and } \beta(\mathbf{x},\mathbf{y})_i = \beta(\mathbf{x}_i,\mathbf{y}_i) \tag{1}$$

Table 1. The alpha (α) and beta (β) operators

$\alpha: A \times A \to B$		
x	**y**	$\alpha(\mathbf{x},\mathbf{y})$
0	0	1
0	1	0
1	0	2
1	1	1

$\beta: B \times A \to A$		
x	**y**	$\beta(\mathbf{x},\mathbf{y})$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Definition 2 (u_β operator). Considering the binary pattern $\mathbf{x} \in A^n$ as input, this unary operator gives the following n-dimensional vector as output.

$$u_\beta(\mathbf{x}) = \sum_{i=1}^{n} \beta(\mathbf{x}_i,\mathbf{x}_i) \tag{2}$$

Definition 3 (Gamma operator). The similarity Gamma operator takes two binary patterns — $\mathbf{x} \in A^n$ and $\mathbf{y} \in A^m$; $n, m \in \mathbb{Z}^+$ $n \leq m$— and a non-negative integer θ as input, and outputs a binary number, according to:

$$\gamma_g(\mathbf{x},\mathbf{y},\theta) = \begin{cases} 1 & \text{if } m - u_\beta[\alpha(\mathbf{x},\mathbf{y}) \bmod 2] \leq \theta \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where mod denotes the usual modulo operation.

The Gamma classifier uses these definitions, as well as the Modified Johnson-Möbius code [4, 5] in order to classify a (potentially unknown) test pattern $\tilde{\mathbf{x}}$, given a fundamental set of learning patterns $\{(\mathbf{x}^\mu,\mathbf{y}^\mu)\}$, by following the next algorithm.

1. Convert the patterns in the fundamental set into binary vectors using the Modified Johnson- Möbius code.
2. Code the test pattern with the Modified Johnson- Möbius code, using the same parameters used for the fundamental set.
3. Compute the stop parameter $\rho = \bigwedge_{j=i}^{n} \bigvee_{i=1}^{p} x_j^i$.
4. Transform the index of all fundamental patterns into two indices, one for their class and another for their position in the class (e.g. $\mathbf{x}^\mu$ in class i becomes $\mathbf{x}^{i\omega}$).
5. Initialize $\theta = 0$.
6. Do $\gamma_g\left(x_j^\mu, \tilde{x}_j, \theta\right)$ for each component of the fundamental patterns.
7. Compute a weighted sum c_i for each class: $c_i = \frac{\sum_{\omega=1}^{k_i} \sum_{j=1}^{n} \gamma_g\left(x_j^\mu, \tilde{x}_j, \theta\right)}{k_i}$, where k_i is the cardinality in the fundamental set of class i.

8. If there is more than one maximum among the different c_i, increment θ by 1 and repeat steps 6 and 7 until there is a unique maximum, or $\theta \geq \rho$.
9. If there is a unique maximum for the c_i, assign $\tilde{y}$ to the class of that maximum.
10. Otherwise, assign $\tilde{y}$ to the class of the first maximum.

4 Prediction Algorithm Based on the Gamma Classifier

The current proposal takes the previous work on univariate time series forecasting and extends it to multivariate prediction, by building the patterns used by the Gamma classifier in a specific way. First, each time series segment is converted into a real-valued vector, by concatenating each data point in the segment. For instance, time series segment D with length l would be converted into vector $\mathbf{v} \in \mathbb{R}^l$:

$$d_1 d_2 \ldots d_l \longrightarrow \mathbf{v}_{l\times 1} = [d_1 \quad d_2 \quad \cdots \quad d_l] \tag{4}$$

Then, any categorical variable is converted to a numeric representation by means of a conversion scale, which should represent the relationships between categories as close to the original as possible. For instance, the categorical feature height could be defined by the set of categories: H_1 = {tall, above-average, medium, below-average, short}, which could be represented by the following scale: H_2 = {19, 18, 17, 16, 15}.

The final patterns are built by concatenating the former vectors, and any converted nominal and numeric variable to be used. Thus, if there are two time series segments (D and E, of length l), two numerical (F and G), and one categorical variable (H), the patterns are built by concatenating the real-valued versions of each variable:

$$D, E, F, G, H \longrightarrow [d_1 \quad d_2 \quad \cdots \quad d_l \quad e_1 \quad e_2 \quad \cdots \quad e_l \quad f \quad g \quad h]_{2l+3} \tag{5}$$

Once the set of patterns has been built and partitioned into fundamental (learning) and testing sets, the Gamma classifier is trained and operated on these sets.

5 Experimental Design

For experiments real production data was compiled for matrix treatments of a naturally fractured Mexican oilfield. Following petroleum engineering practice, the effect of a treatment is measured during the three months after it. Since matrix treatments affect all fluids from the porous media, oil production Q_o, water production Q_w, and gas/oil ratio (GOR) after treatment are predicted. Since we consider six months before the treatment as production input data, the available data was filtered such that any interference of other treatments during the observed 9 months are excluded, leaving only 43 cases which were recorded at 24 producing wells.

One of the first steps of preprocessing for data mining is rescaling, since input parameters are often of different units and scales. Since both of the rescaling techniques (normalization and standardization) have their known drawbacks, in order to compare the behavior of the Gamma classifier, experiments were run for real, normalized and standardized data. The discussion of these results is out of the scope of this paper.

Since the prediction with normalized data set showed slightly better accuracy, we refer to this setting in further discussion, while de-normalized data are used for the analysis of the results.

As validation technique for the experiments, a Leave-One-Out (LOO) cross-validation is used, in which the model is repeatedly refit leaving out a single observation and then used to derive a prediction for the left-out observation. Although LOO gives prediction error estimates that are more variable than other forms of cross-validation such as *K*-fold (in which the training/testing is performed after breaking the data into *K* groups), LOO is chosen due to the small size of available data. The obtained results are compared with other two prediction methods commonly used in DM and petroleum engineering: the nearest-neighbor method and Bayesian networks.

The nearest-neighbor method is perhaps one of the simplest of all algorithms for prediction, and is called the *k*-NN algorithm. For the experiments the $k = 1$ version (IB1) is selected. Bayesian network (BN) and Naïve BN models were also used for comparison of the results of our approach. This selection is explained by the commonly observed effectiveness of the Bayesian methods for modeling situations where some prior information is known and incoming data are uncertain or partially unavailable. A particular advantage of BNs over predictive models based on neural networks is that BN structure explicitly represents the inter-relationships among the data-set attributes, which favors their acceptance in petroleum engineering practice [9].

To provide the experiments, the STATISTICA software was used for feature selection, the Gamma classifier based proposal was implemented in MATLAB and the reference DM models were selected from the WEKA open source DM tool.

6 Experimental Results and Their Analysis

Since the methods used for comparison on the experimental data (IB1, NaïveBayes, and BayesNetwork) do not lend themselves to naturally work with time series, the input patterns (initially made up by three time series segments 6 samples long each, 6 numeric, and one categorical variable), were converted into input patterns of 9 numeric and one categorical variables. Meanwhile, the output patterns (originally 3 time series of 3 points each), became 3 numerical variables. All numeric variables — including the converted time series— have been normalized to values between 0 and 1 inclusive, depending on the minimum and maximum values for each feature, at each particular record. Given the characteristics of the Gamma classifier, the proposed model is able to deliver the output patterns directly. However, the methods used for comparison require a post-processing stage, consisting of a look-up table matching class and corresponding output pattern.

Table 2 shows the prediction performance on the data set with average data instead of time series, expressed in Mean Square Error (MSE) for each output variable. The Gamma classifier offers competitive performance: the error for Q_o is the best among the four models, while that for GOR is below the mean, although it presents the larger error for Q_w. BayesNetwork shows the best error on GOR, while its Q_o error is the

Table 2. Prediction performance using summarized data (average), in MSE. The best performers are shadowed.

Model	Q_o	Q_w	GOR
BayesNet	3.99E+7	9.48E+6	1.68E+3
Gamma	3.06E+7	3.44E+7	7.56E+3
IB1	3.70E+7	7.20E+6	1.81E+4
NaïveBayes	3.95E+7	2.16E+7	4.42E+3

worst. However, considering that oil is usually the variable of most interest, and that in practice, monthly production forecasts are a common requirement, the prediction of monthly production of oil, water, and gas are necessary.

To address the latter, a second data set was built, using the original time series instead of their summarized versions, along with the pattern building method presented above for the Gamma classifier, lending part of the proposal robustness to the other models. Thus, the second data set has input patterns of 25 dimensions built from the 3 time series segments, 6 numeric, and one categorical variable; while the output patterns are 9-dimensional vectors built from the 3 time series segments.

The summarized results for this second experiment can be seen in Table 3. The proposed model is competitive again, performing much better than before. The results shown by the Gamma classifier are below the mean for eight of the nine output variables, being the best result for Q_o2 and Q_o3, even though Q_o1 exhibits a larger error than BayesNet. NaïveBayes shows the worst performance for the experimental data set, giving the largest error for 6 of the 9 output variables, even when its performance for Q_o3 is below the mean. Also, notice that BayesNet offers the best errors for the three variables related to water: Q_w1, Q_w2, and Q_w3.

Table 3. Prediction performance on the second experiment, in MSE. Results below the mean are shadowed

Model	Q_o1	Q_o2	Q_o3	Q_w1	Q_w2	Q_w3	GOR1	GOR2	GOR3
Gamma	5.01E+7	5.70E+7	4.62E+7	4.06E+7	3.48E+7	3.34E+7	6.82E+3	1.40E+4	1.32E+4
IB1	5.57E+7	6.77E+7	1.28E+8	8.59E+7	5.63E+6	8.69E+6	2.75E+3	4.24E+3	3.19E+4
BayesNet	3.48E+7	9.51E+7	1.01E+8	6.29E+6	8.41E+6	1.29E+7	2.53E+4	2.62E+4	3.72E+4
NaïveBayes	5.64E+7	8.60E+7	8.80E+7	1.15E+8	1.21E+8	1.14E+8	3.12E+4	3.08E+4	3.06E+4

Table 4. Prediction performance of the Gamma classifier on the second experiment, in SMAPE

Model	Q_o1	Q_o2	Q_o3	Q_w1	Q_w2	Q_w3	GOR1	GOR2	GOR3
Gamma	7%	12%	12%	24%	33%	33%	8%	10%	9%

Water production presents difficulties for all models, and the Gamma classifier in particular. If we select as an optimality measure the adjusted or symmetric MAPE (SMAPE), the proposal shows the following results (Table 4). This may be due to its high variability: mean is 2288.53 while standard deviation is 4899.78.

In general, a SMAPE of 10% is considered to be very good. Also, for a similar problem (candidate well selection for gel treatment), the methods based on neural and Bayesian networks reported the average accuracy in predictions from 77 to 84% [9, 10]. They were tested on data sets from 22 and 14 wells respectively. For the ANN model used for the estimation of monthly oil production potential, the correlation coefficient between predicted and actual production was 0.79 [11]. They all used summarized numeric variables. As it can be seen from Table 4, the use of time series instead of averaged productions increase the accuracy up to 90% (obviously the comparison is relatively unfair since the experimental settings were different).

These results emphasize some of the Gamma classifier benefits. Even though it does not need to summarize time series data into a numeric value to work with them (as the other methods do) the proposal offers competitive performance on the first experiment, even giving the lowest error for oil production prediction, which is usually the fluid of most interest. On the other hand, when working with the original time series segments coded as proposed —and even lending this coding technique to the other methods— our proposal offers a better, more robust performance, with 8 of 9 output variables below their means, and two lowest errors (again for oil). This consistent behavior is of particular interest, given that in practice a method that gives good results (even when they are sub-optimal) is quite appreciated.

Notice also that these competitive performances are obtained without greater complexity. As discussed in [4], the Gamma classifier has a complexity of $O(np)$ for a fundamental set of p patterns $\mathbf{x}^{\mu} \in \mathbb{R}^n$, while the complexity of the simpler of the other models, k-NN, is also $O(np)$. However, given the small size of the data sets used for the experiments, the differences in complexity among the models is barely noticeable in execution time, even against the most complex one: the BayesNet; these experiments are run in less than 3 seconds, for any models used.

7 Conclusions

An extended version of Gamma classifier combined with a new coding technique have been developed to predict post-treatment well behavior, based on future values of oil and water production along with GOR after matrix treatments. Compared to several well-known prediction techniques (Bayesian network, Naïve Bayes and the k-NN algorithm) the experimental results show a competitive performance by the Gamma classifier.

As experimental results show, the Gamma classifier outperforms all the competitors in half of the cases. Yet, its main advantage is the balanced behavior in prediction: for almost all the cases it shows above average accuracy, even when its performance is sub-optimal. Another of its advantages is that compared to its competitors it doesn't need pre-processing: it generates real vectors of arbitrary dimension. In order to apply the proposal to different data, such as hydraulic fracturing, gel treatment, or at different time intervals, new fundamental and test pattern sets should be built and presented to the method. Currently the model is used in decision support systems for candidate well selection for matrix treatment and water control.

Acknowledgements. The authors would like to thank Dra. Ana Cosultchi for her help in definition and pre-processing of data samples. Partial financial support for this work has been provided by CONACYT-SENER (project 146515), as well as the Instituto Politécnico Nacional (Secretaría Académica, COFAA, SIP, CIDETEC, and CIC), CONACyT, and SNI. The first author would like also to thank the IMP for postdoctoral fellowship.

References

1. Hand, D., Manilla, H., Smyth, P.: Principles of Data Mining. MIT Press, Cambridge (2001)
2. Batyrshin, I., Sheremetov, L., Markov, M., Panova, A.: Hybrid Method for Pore Structure Classification in Carbonate Formations. J. of Petroleum Science and Engineering, Elsevier Science 47(1-2), 35–50 (2005)
3. Zhao, D., Yan, J.: Performance prediction methodology based on pattern recognition. Signal Processing 91(9), 2194–2203 (2011)
4. López-Yáñez, I., Arguelles-Cruz, A.J., Camacho-Nieto, O., Yáñez-Márquez, C.: Pollutants Time-Series Prediction Using the Gamma Classifier. Int. J. of Computational Intelligence Systems 4(4), 680–711 (2011)
5. López-Yáñez, I., Sheremetov, L., Yáñez-Márquez, C.: Associative model for the forecasting of time series based on the gamma classifier. In: Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Rodríguez, J.S., di Baja, G.S. (eds.) MCPR 2013. LNCS, vol. 7914, pp. 304–313. Springer, Heidelberg (2013)
6. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons, New York (2001)
7. Bolt, J.H., van der Gaag, L.C.: Multi-dimensional classification with naive Bayesian network classifiers. In: Belgian/Netherlands Artificial Intelligence Conference (2012)
8. Seright, R.S., Lane, R.H., Sydansk, R.D.: A Strategy for Attacking Excess Water Production. SPEPF 18(3), 158–169 (2003), doi:10.2118/84966-PA
9. Ghoraishy, S.M., Liang, J.T., Green, D.W., Liang, H.C.: Application of Bayesian Networks for Predicting the Performance of Gel-Treated Wells in the Arbuckle Formation, Kansas. Paper SPE 113401 presented at the 2008 SPE Improved Oil Recovery Symposium held in Tulsa, Oklahoma, U.S.A, April 19-23 (2008)
10. Saeedi, A., Camarda, K.V., Liang, J.T.: Using Neural Networks for Candidate Selection and Well Performance Prediction in Water-Shutoff Treatments Using Polymer Gels- A field Study. Paper SPE 101028 presented at the 2006 SPE Asia Pacific Oil & Gas Conference and Exhibition, Adelaide, Australia, September 11-13 (2006)
11. Schrader, S.M., Balch, R.S., Ruan, T.: Using Neural Networks to Estimate Monthly Production: A Case Study for the Devonian Carbonates, Southeast New Mexico. In: SPE Production Operations Symposium, Oklahoma, USA, SPE 94089, April 16-19 (2005)

On Preference Order of DRSA Conditional Attributes for Computational Stylistics

Urszula Stańczyk

Institute of Informatics, Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland

Abstract. Computational stylistics (or stylometry) advocates that any writing style can be uniquely defined by quantitative measures. The patterns observed in textual descriptors are specific to authors in such high degree that it enables their recognition. In processing there can be employed techniques from artificial intelligence domain such as Dominance-Based Rough Set Approach (DRSA). Its starting point comprises establishing a preference order in value sets of attributes, which can be dictated by domain knowledge, arbitrarily assumed, or obtained through some analysis. The paper describes the process of finding preference orders through such sequential adjustments that lead to induction of minimal cover decision algorithms with the highest classification accuracy.

Keywords: Computational Stylistics, Authorship Attribution, DRSA, Decision Algorithm, Preference Order, Conditional Attributes.

1 Introduction

Authorship of handwritten texts is usually recognised through analysis of shapes of letters, slant, spaces between characters. When a text is prepared using some word processor, all these descriptive features cease to be individual as they are applied automatically. Yet also for manuscripts in electronic formats analysis leading to authorship attribution can be performed, by employing the concept of writer or author invariant that is fundamental to computational stylistics [3].

Computational stylistics is a study of writing styles that relies on numerical characteristics of texts [2]. Descriptors used refer to lexical, syntactic, content-specific, or structural features, which give statistics of usage for characters, words, reflect organisation of a text, its content, formatting. When markers allow for recognition of authors, they correspond to writer invariants. The process can be treated as detection and recognition of patterns [1], hence in data analysis some connectionist or rule-based classifiers can be employed. Dominance-Based Rough Set Approach constitutes an example from this last group [7].

DRSA procedures, proposed to support multicriteria decision making [4], enable both nominal and ordinal classification by observing preference orders in value sets of conditional attributes. Depending on the application area, preference orders can be naturally determined by general or domain knowledge, yet

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 26–33, 2013.

in its absence some approach must be employed. One way to proceed is firstly to make an arbitrary assumption then optimise induced decision algorithms [9], another requires establishing an order by some trial and error procedures. This latter strategy was implemented in the research presented in this paper.

From stylometric perspective, not only textual markers and their values comprise authorial invariants but preference orders present in the value sets as well, and as such by definition they are task-dependent. Therefore, by adapting the sequential search there was proposed an algorithm for fine-tuning preference orders of DRSA conditional attributes that results in induction of minimal cover decision algorithms while keeping or increasing their classification accuracy.

The process starts with assumption of a preference order for all attributes. Next a preference order of a single attribute is changed and decision algorithm calculated, iteratively for all attributes, and there is chosen the one with the highest classification accuracy. The conditional attribute whose change lead to selection is excluded from the considered set and the whole procedure is repeated. The processing stops when it is impossible to find any new preference order with higher quality of approximation, when either all attributes are already considered, or all changes bring worsening of performance. Since the search procedure is not exhaustive, it does not guarantee finding the absolute best preference order, yet it gives a reasonable chance at obtaining such.

The paper is organised as follows. In Section 2 some fundamental notions of computational stylistics are presented. Section 3 describes elements of rough set theory while Section 4 details the experimental setup. Algorithm for establishing preference order for conditional attributes is explained in Section 5 along with results obtained from tests. Concluding remarks are given in Section 6.

2 Fundamental Notions of Computational Stylistics

To prove or disprove the authenticity or settle the questions about disputed texts, in its early days textual analysis relied on striking features. Since these kinds of descriptors can be imitated, falsification of someone else's style was not out of realm of possibility. Modern word processors make possible forgery even easier yet computational powers of computer technologies enable also much more complex text processing, detect characteristics invisible to the bare human eye, and observe subtle linguistic elements, employed in less conscious way.

Computational stylistics studies documents through quantitative measures which allow for characterisation of authors as well as detect differences and similarities among texts. The fundamental claim states that, given representative number of writing samples, it is possible to compile such set of textual markers that enables unambiguous recognition of authorship. It is called writer or author invariant. By definition author invariants capture individuality of writing styles, so there is no general arbitrary procedure as to how to construct them.

Markers that are widely used in writer recognition refer to lexical and syntactic characteristics, the former giving frequencies of usage for letters, words, averages of word length, sentence length, etc., while the latter express the general composition of sentences, paragraphs, as lined out by the punctuation marks [6].

A choice of a processing technique is yet another essential point for considerations, even more important when taking into account that some methodologies determine also the type of data sets they work upon. Furthermore, data mining procedures quite often possess their own mechanisms dedicated to detection of importance for studied features. As a result, the popular approach is to consider descriptors within the context of a particular data processing technique employed, treating them as not only task- but methodology-dependent as well.

Techniques used involve either statistic-oriented computations or artificial intelligence methodologies [5]. DRSA belongs in this latter group.

3 Dominance-Based Rough Set Approach

Theory of rough sets is dedicated to cases of imperfect and incomplete knowledge, with information about the objects of the universe U seen in the form of granules. In Classical Rough Set Approach, defined by Z. Pawlak in 1982 [5], these granules are determined by the indiscernibility relation that returns equivalence classes of objects that cannot be discerned, basing on which CRSA observes presence or absence of features and enables nominal classification by rough approximations.

The data is contained in the decision table DT, an information system:

$$DT = (U, Q) = (U, C \cup D) \tag{1}$$

where Q is a finite set of attributes describing objects of the universe U, divided into C conditional and D decision attributes. An irreducible subset of conditional attributes that maintains the quality of approximation of the whole decision table is called a relative reduct. For all attributes included in Q there exist functions f_q, determining $f_q : U \rightarrow V_q$ with V_q being the set of values of attribute q.

When the values of attributes show monotonic properties the indiscernibility relation can be insufficient for efficient multicriteria decision making and that is why it has been proposed to replace it with dominance which resulted in Dominance-Based Rough Set Approach (DRSA) [4,7]. Observation of preference order for attributes allows for ordinal classification.

If for all criteria $q \in P \subseteq C$, $x \succeq_q y$ ($\succeq_q$ stands for a weak preference relation), then x dominates y with respect to P, denoted by xD_Py. $D_P^+(x)$ is a set of objects dominating x, and $D_P^-(x)$ a set of objects dominated by x. These two sets define the granules of knowledge in DRSA as follows:

$$D_P^+(x) = \{y \in U : yD_Px\} \qquad D_P^-(x) = \{y \in U : xD_Py\} \tag{2}$$

The preference order for the decision attribute $D = \{d\}$ is determined by its values which partition the universe into decision classes $\boldsymbol{Cl} = \{Cl_t\}$, for $t = 1, \ldots, K$. The increasing indices indicate increasing preference of decision classes. The sets of objects to be approximated are upward or downward unions of these classes (dominance cones):

$$Cl_t^{\geq} = \bigcup_{s \geq t} Cl_s \qquad Cl_t^{\leq} = \bigcup_{s \leq t} Cl_s \tag{3}$$

P-lower approximation of $Cl_t^{\geq}$ is the set of objects that belong to $Cl_t^{\geq}$ (denoted as $\underline{P}(Cl_t^{\geq})$), and P-upper approximation of $Cl_t^{\geq}$, $\overline{P}(Cl_t^{\geq})$, is the set of objects which could belong to $Cl_t^{\geq}$:

$$\underline{P}(Cl_t^{\geq}) = \{x \in U : D_P^+ \subseteq Cl_t^{\geq}\} \qquad \overline{P}(Cl_t^{\geq}) = \{x \in U : D_P^- \cap Cl_t^{\geq} \neq \emptyset\} \quad (4)$$

Finding approximations of the dominance cones constitutes the starting point for the procedure of induction of decision rules, which can be certain, possible, or approximate. A set of decision rules is complete when no object of the decision table remains unclassified. It is minimal when it is complete and irreducible.

The minimal set of rules does not guarantee the highest classification accuracy, which is typically considered as the most important factor indicating how good the algorithm is, as there are found only the rules sufficient to cover the learning samples. Especially in case of real-valued attributes these samples cannot cover all points of the multidimensional input space and the testing set could be so different that it results in incorrect recognition or no rules matching. So, there are also tried strategies with inducing all rules and then construction of optimised classifiers by selection of rules based on measures of importance [8].

4 Experimental Setup

To achieve reliability, the markers have to be calculated over representative number of samples. In fact, the bigger the corpus, the higher chance at good recognition ratio. That is why in the experiments there were processed selected literary works of two writers, Edith Wharton and Jane Austen. Samples for the training and testing set were constructed basing on chapters from the selected novels.

For authorship attribution many candidate sets of characteristic features are proposed [3]. In the research there were studied frequencies of usage for:

- 8 punctuation marks: a fullstop, a comma, a question mark, an exclamation mark, a semicolon, a colon, a bracket, a hyphen.
- The lexical markers arbitrarily selected basing on the list of the most popular words in English language: but, and, not, in, with, on, at, of, this, as, that, what, from, by, for, to, if.

It resulted in the set with the total cardinality equal 25, successfully used in the past experiments [9]. Once the frequencies for all markers were calculated for all samples, the obtained data sets were used to construct the decision table.

Frequencies of usage are continuous real values with natural ordering, but DRSA requires definition of *preference* order which is more demanding. From the stylometric perspective it is not possible to state *how* these attributes are preference ordered as it would imply the existence of some a priori knowledge that some frequencies point to specific writers. On the other hand, we do know that lexical and syntactic descriptors enable authorship attribution, which in turn does imply that certain combinations of calculated frequencies are associated with certain authors. This observation confirms the existence of preference orders and permits to employ DRSA in text mining.

Unfortunately, there is no general arbitrary methodology for establishing preference orders for all attributes, which would be applicable regardless of the studied area. Usually in such cases either there is assumed some arbitrary order or preferences are adjusted by some trial and error procedures.

In the preliminary stage of experiments two opposite preference orders have been assumed, the same for all conditional attributes, and the minimal cover decision algorithms generated. The performance of these classifiers was then verified with testing. The results are given in three categories of decisions: correct, incorrect, and ambiguous. The third category is used in case of no rules matching a sample, or when there are several matching rules with contradicting decisions. All situations of ambiguous decisions were treated as incorrect.

The minimal cover algorithm calculated for decreasing preference order assumed for conditional attributes without any constraints on rules recognised barely half of the testing samples. Standard procedure helps in limiting the rules by imposing some hard constraints upon their support or length [9]. Support of a rule states for how many training samples it is valid while length specifies the number of elementary conditions included in the premise part of the rule. No limitations on rules did, however, increase the performance of this classifier.

For increasing preference order of all attributes the minimal cover algorithm returned 62% recognition while without constraints, which increased to 76% when only decision rules with supports equal or higher than 6 were taken into consideration. This result can be treated as satisfactory, but it can still be enhanced by fine-tuning preference orders of attributes.

5 Algorithm for Establishing Preference Order

When generation of all rules on examples decision algorithm and its optimisation is considered as too time-consuming, and minimal cover algorithm gives unsatisfactory classification results, yet another strategy can be employed focusing on preference orders of attributes, adjusting them to the specifics of a task. The strategy is illustrated with the previously described problem of authorship attribution with 25 conditional attributes.

The conditional attributes reflect naturally ordered frequencies of usage of lexical and syntactic markers. From the point of DRSA an order can be treated as either increasing or decreasing (the higher value the higher or the lower class, respectively). For $N = 25$ attributes, the total number of possible orderings is $2^N = 2^{25}$. The exponential character of this relation makes processing all of them in order to find the best unfeasible. Instead there is proposed another systematic approach, listed in Algorithm 1, which by adapting the sequential search limits a number of preference orders to be checked to more manageable polynomial:

$$N + (N-1) + (N-2) + \cdots + 2 + 1 = \frac{(N+1)N}{2} \tag{5}$$

in the worse case scenario, when through iterative introduction of changes for a single attribute at a time the complete set of features is considered and at the

end all attributes have the preference orders opposite to the ones they had at the beginning. The execution of the algorithm can be stopped earlier if all new changes result only in worsening of performance.

Algorithm 1. Algorithm for establishing preference orders of conditional attributes based on analysis of classification accuracy of minimal cover decision algorithms generated, with changes of preference orders for single attributes

```
Input:  Decision Table DT, number of conditional attributes N
Output: Best preference order BestPrefOr of conditional attributes
        Best minimal cover decision algorithm BestMCDA
Begin
        for each conditional attribute CondAttr do
            assume a preference order
        endfor
        BestPrefOr ← current preference order PrefOr
        generate MCDA
        BestCA ← ClassifAccuracy(MCDA)
        BestMCDA ← MCDA
        NrCondAttrLeft ← N
        repeat
            IndxAttr ← 0
            for i = 1 to NrCondAttrLeft do
                change preference order of CondAttr(i)
                generate MCDAi
                if ClassifAccuracy(MCDAi)≥ BestCA then
                    BestCA ← ClassifAccuracy(MCDAi)
                    BestMCDA ← MCDAi
                    BestPrefOr ← PrefOri
                    IndxAttr ← i
                else disregard change of preference order of CondAttr(i)
                endif
            endfor
            NrCondAttrLeft ← NrCondAttrLeft-1
            if IndxAttr>0 then
                remove CondAttr(IndxAttr) from the set
            endif
        until (NrCondAttrLeft=0) OR (IndxAttr=0)
End
```

At the beginning, the minimal cover algorithm for all attributes with the same preference order is taken as the reference point. Next, going through the complete set, a preference order of a single attribute is changed and minimal cover algorithm calculated. From all $N = 25$ orderings there is selected the one for which classification accuracy is the highest and at least the same as the one previously obtained. The attribute whose change resulted in the increase of performance is excluded from further considerations and the higher recognition ratio taken as the new reference point. The procedure is repeated for the gradually decreasing

($N-1=24$, then $N-2=23$, and so on) number of attributes until the stopping condition is satisfied when either no attribute remains yet to be considered, or when all most recent changes bring only decreasing performance.

The search for preference order does not try exhaustively all possible orderings, so it cannot guarantee to result in the best solution. The search space could be expanded by repeating the procedure with different staring points, with some arbitrary or random selection of starting preference orders for attributes. When we can afford to explore more search paths, we can choose the best from them, yet even with following one path in the binary tree of changes offers a reasonable chance at arriving at satisfactory results, which is illustrated in Table 1.

Table 1. Establishing the preference order of conditional attributes by sequential search. The 0th stage of execution is a starting point where all attributes have decreasing order, in subsequent stages changed into increasing. The stage with index Z is the opposite starting point with all attributes of increasing preference, next changed into decreasing. At each stage attributes adjusting preference order are indicated and classification accuracies for the best generated minimal cover decision algorithms given.

		→					Stage of execution							←	
	0	1	2	3	4	5	6	7	8	U	V	W	X	Y	Z
Conditional Attribute		not	what	:	at	in	to	but	,	-	on	!	and	;	
Classification accuracy [%]	51	63	74	75	80	84	84	85	84	77	82	82	81	73	62

In the first group of experiments the preference order adjustment procedure stopped at the 8th stage. The set of conditional attributes is not yet exhausted, but all modifications within this stage resulted in worsened performance. The best algorithm at this stage has classification accuracy of 84% while at the previous 7th stage the predictive accuracy was 85%. Starting from the opposite end, with all conditional attributes assigned to increasing preference order and then changing them one by one to decreasing, resulted in execution of the procedure for just 5 times. At the Vth stage the calculated algorithm returned 82% recognition ratio that was only worsened by all further changes.

Within the proposed approach

25+24+23+22+21+20+19+18=172 and

25+24+23+22+21=115

(the total of 287) out of 2^{25} possible preference orderings were tested. Obviously, there is no guarantee that the two best preference cases found are the best in general, as these maximal classification accuracies could be, and usually are rather local than global, yet the presented algorithm gives a reasonable chance to arrive at more informed choice of preference orders than assumed arbitrarily, when otherwise there is no domain knowledge to support this choice.

6 Conclusions

The paper presents an algorithm for establishing preference orders of conditional attributes within Dominance-Based Rough Set Approach to the stylometric task of authorship attribution. For the considered lexical and syntactic characteristic features referring to usage frequencies of textual markers, the real-valued data sets are naturally ordered. However, whether the monotonicity is perceived as increasing or decreasing depends on perspective which betrays individuality of a writing style. Since domain knowledge is insufficient to define such preferences and arbitrary assumptions do not necessarily result in satisfactory performance of the constructed classifiers, by adapting the sequential search there was proposed a procedure for iterative fine-tuning of preference orders for conditional attributes by observing how changes of preference of single features influence the classification accuracy of the generated minimal cover decision algorithms. Even though it is not exhaustive, the search process offers a reasonable chance at finding the solution that satisfies requirements.

Acknowledgements. Texts used are available thanks to Project Gutenberg (`http://www.gutenberg.org`). 4eMka Software employed for DRSA processing [4,7] was developed at the Laboratory of Intelligent Decision Support Systems, (`http://www-idss.cs.put.poznan.pl/`), Poznan University of Technology.

References

1. Abraham, A., Falcón, R., Bello, R. (eds.): Rough Set Theory: A True Landmark in Data Analysis. SCI, vol. 174. Springer, Berlin (2009)
2. Argamon, S., Burns, K., Dubnov, S. (eds.): The structure of style: Algorithmic approaches to understanding manner and meaning. Springer, Berlin (2010)
3. Craig, H.: Stylistic analysis and authorship studies. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) A Companion to Digital Humanities. Blackwell, Oxford (2004)
4. Greco, S., Matarazzo, B., Slowinski, R.: Rough set theory for multicriteria decision analysis. European Journal of Operational Research 129(1), 1–47 (2001)
5. Pawlak, Z.: Rough sets and intelligent data analysis. Information Sciences 147, 1–12 (2002)
6. Peng, R., Hengartner, H.: Quantitative analysis of literary styles. The American Statistician 56(3), 15–38 (2002)
7. Słowiński, R., Greco, S., Matarazzo, B.: Dominance-based rough set approach to reasoning about ordinal data. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 5–11. Springer, Heidelberg (2007)
8. Stańczyk, U.: Decision rule length as a basis for evaluation of attribute relevance. Journal of Intelligent and Fuzzy Systems 24(3), 429–445 (2013)
9. Stańczyk, U.: Rough set and artificial neural network approach to computational stylistics. In: Ramanna, S., Jain, L., Howlett, R. (eds.) Emerging Paradigms in Machine Learning. SIST, vol. 13, pp. 441–470. Springer, Heidelberg (2013)

Entity Matching Technique for Bibliographic Database

Sumit Mishra, Samrat Mondal, and Sriparna Saha

Department of Computer Science and Engineering,
Indian Institute of Technology Patna, India
{sumitmishra,samrat,sriparna}@iitp.ac.in

Abstract. Some of the attributes of a database relation may evolve over time i.e., they change their values at different instants of time. For example, *affiliation* attribute of an author relation in a bibliographic database which maintains publication details of various authors, may change its value. When a database contains records of this nature and number of records grows to a large extent then it becomes really very challenging to identify which records belong to which entity due to lack of a proper key. In such a situation, the other attributes of the records and the timed information associated with the records may be useful in identifying whether the records belong to the same entity or different. In the proposed work, the records are initially clustered based on *email-id* attribute and the clusters are further refined based on other temporal and non-temporal attributes. The refinement process involves similarity check with other records and clusters. A comparative analysis with two existing systems DBLP and ArnetMiner shows that the proposed technique can able to produce better results in many cases.

Keywords: Bibliographic Database, Entity Matching, Temporal Data, Similarity Check.

1 Introduction

Entity matching technique identifies records that refer to the same entity. It is a well known problem in the field of database and artificial intelligence. Other synonyms that are used to refer to this problem are merge/purge problem [5], object distinction [10], temporal record linkage [6,7], author name ambiguity [8], etc,. Researchers have proposed various approaches to tackle this problem. Among the available techniques two broad classes can be identified. The first one deals with non-temporal attributes only. Relationships among records are established by using different rule based strategies like [5,2,3,9] or machine learning based strategies like [1,10].

In addition to non-temporal attributes, second one deals with temporal attributes [4] also. Many bibliographic databases like DBLP contain temporal data, for example, the *month* and *year* of published papers. Often such temporal information provides additional evidences in linking records. However, the importance of such temporal attribute in entity matching technique is only considered

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 34–41, 2013.

recently [6,7] where authors tried to link the records based on the concepts of *time decay* which is used to capture the effect of elapsed time on entity value evolution. The attributes considered here are *name*, *affiliation* and *co-author*. However, attributes like *email id*, *venue*, *reference* are ignored here although they provide some additional clue in merging records.

In this paper, we have proposed an entity matching technique for bibliographic database which is also termed as EMTBD. We have considered the temporal attribute like *time-stamp* of the paper along with several other non-temporal attributes like *author*, *co-author*, *affiliation* of author and co-author, *email-id* of author, *venue* and *references*. Similarity check and clustering techniques are used to obtain the result. For similarity computation, different threshold values are used. The threshold values are computed dynamically and depend on the number of records available for a particular author. The proposed EMTBD technique is compared with DBLP[1] and ArnetMiner[2] which maintain a database of large number of research papers.

The rest of the paper is organized as follows: Section 2 gives a description of proposed methodology. Section 3 emphasizes on RAC-Cluster which is the primary module of our approach. In section 4 we have detailed different datasets and performed analysis on our results. Finally, Section 5 concludes the paper and gives the future direction of the work.

2 Proposed Methodology

The proposed EMTBD technique is described using Figure 1 and Figure 2. The steps of the proposed methodology is illustrated as follows.

2.1 Initial Classification

We start with creating clusters based on *email-ids* of the authors. The reason is that it is very rare for two authors to share the same *email-id*. This classification leads to two types of clusters - (1) *Cluster with email-id* and (2) *Cluster without email-id*.

2.2 Affiliation Email Cluster (AE-Cluster)

In this step, the clusters in *cluster with email-id* are processed. Two such clusters are merged if they have the common *affiliation* and same *initials of email-id*.

2.3 Affiliation Co-author Cluster (AC-Cluster)

Here we process the clusters in *cluster without email-id*. We merge these types of clusters if they have the same *affiliation* and some common *co-author*. For this we have used *Co-authorship graph* which we create for each *affiliation*.

[1] `http://www.informatik.uni-trier.de/ ley/db/`
[2] `http://arnetminer.org/`

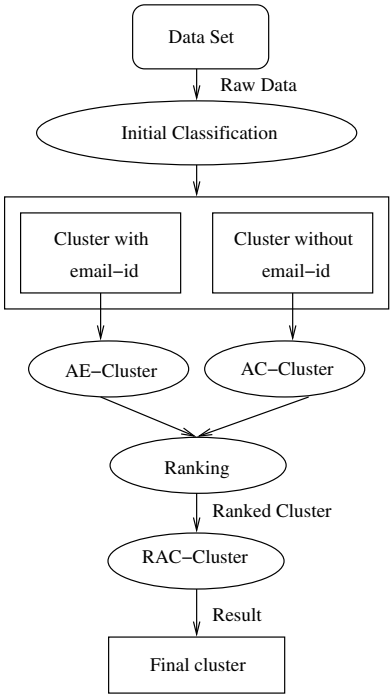


Fig. 1. System Description

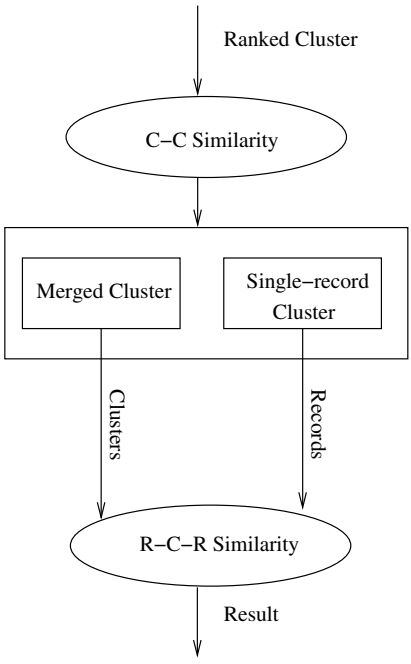


Fig. 2. RAC-Cluster

Co-authorship Graph: It is an undirected graph $G = (V, E)$ such that: V is a set of vertices, where each $v_i \in V$ denotes the distinct *co-author* of an *affiliation*. E is a set of edges, where each $e \in E$, represents an edge between two vertices u and v if and only if the co-authors represented by these vertices belong to the same record.

Let's illustrate the co-authorship graph with an example. Table 1 shows six clusters with their *co-authors* and *affiliation*. The *co-authorship graph* of these clusters shown in Figure 3 provides two component: $\{C_1, C_2, C_3, C_4\}$ and $\{C_5, C_6\}$. These components determine the number of resultant clusters that are obtained from the given clusters. Here six clusters are merged into two clusters as described by the above two components.

2.4 Ranking

The clusters obtained after AE-Cluster and AC-Cluster are then considered together and a rank is assigned to each cluster. The ranking process starts with arranging the records in the order of non-decreasing *time-stamp*. Two notations-*earliest time-stamp* or ETS and *latest time-stamp* or LTS are introduced here. ETS is the oldest *time-stamp* among all the records of the cluster under consideration while LTS is the recent most *time-stamp* among all the records of the

Table 1. Clusters C_1, C_2, $\cdots$, C_6 with their co-authors and affiliation

Cluster	Co-author	Affiliation
C_1	ca_1,ca_2	xyz
C_2	ca_2,ca_3	xyz
C_3	ca_5,ca_6	xyz
C_4	ca_3,ca_4,ca_5	xyz
C_5	ca_7,ca_8	xyz
C_6	ca_8,ca_9	xyz

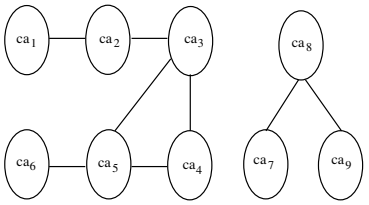


Fig. 3. Co-authorship graph of the clusters shown in Figure

cluster under consideration. ETS and LTS may be same in case the cluster has single record or all the records of the cluster have the same *time-stamp*.

For ranking module, we assign rank 1 to that cluster having minimum ETS. If two or more clusters are having the same ETS then assign the rank to that cluster having lowest LTS. This whole process is repeated until rank is assigned to each clusters.

Table 2. Clusters in the increasing *time-stamp* order of their records

C_1	C_2	C_3	C_4
Jan-1999	Feb-1999	Feb-1999	Feb-2000
May-1999	Feb-2000	Feb-2001	Mar-2000
Jan-2001	Feb-2001	Sep-2001	Jul-2000
Sep-2002	May-2002	Oct-2001	Nov-2000

Table 3. ETS and LTS of different clusters with their rank shown in Table 2

Cluster	ETS	LTS	Rank
C_1	Jan-1999	Sep-2002	1
C_2	Feb-1999	May-2002	3
C_3	Feb-1999	Oct-2001	2
C_4	Feb-2000	Nov-2000	4

In Table 2, four clusters C_1, C_2, C_3 and C_4 are shown. Each record of the cluster is associated with a *time-stamp*. ETS and LTS of each cluster together with their rank are shown in Table 3. The basic aim of assigning rank to clusters is that rank arranges them in *time-stamp* order and reduces the no. of steps for merging of two clusters.

3 RAC Cluster

RAC Cluster module is depicted in Figure 2. It is termed as RAC because here we process the records(R) and clusters(C) using various attributes(A). This module is used to process the ranked clusters in two steps.

3.1 C-C Similarity Based Clustering

C-C Similarity is calculated between two clusters. At first we check whether two clusters are intersecting in nature. To understand this, let et_i and et_j denote

the ETS of the cluster C_i and C_j respectively while lt_i and lt_j represent the corresponding LTS. Formally, two clusters C_i and C_j are **intersecting** if any one of the following condition satisfies: $(1) et_j \leq et_i \leq lt_j$ $(2) et_j \leq lt_i \leq lt_j$ $(3) et_i \leq et_j \leq lt_i$ $(4) et_i \leq lt_j \leq lt_i$

We now introduce two terms *Time Line* and *Variation-count* to analyse the clusters further.

Time Line: It is a line joining the points representing the *time-stamps* of each record of the cluster. The *time-stamps* can be represented in a time dimension-axis. Let $T_1 = \{t_{1a}, t_{1b}, \cdots, t_{1k}\}$ and $T_2 = \{t_{2a}, t_{2b}, \cdots, t_{2m}\}$ be the sets of points representing the *time-stamp* of each record in clusters C_1 and C_2 respectively. k and m are no. of records in clusters C_1 and C_2 respectively. Using sets T_1 and T_2, two *Time Line* can be obtained.

Now a *variation-count* parameter is introduced to check the no. of times *author* has changed his/her *affiliation* between two *affiliation*. Consider the point representing min(ETS(T_1), ETS(T_2)). From this point, draw a line to next nearest *time-stamp* point. The process continues till the point representing max(LTS(T_1), LTS(T_2)) is not reached.

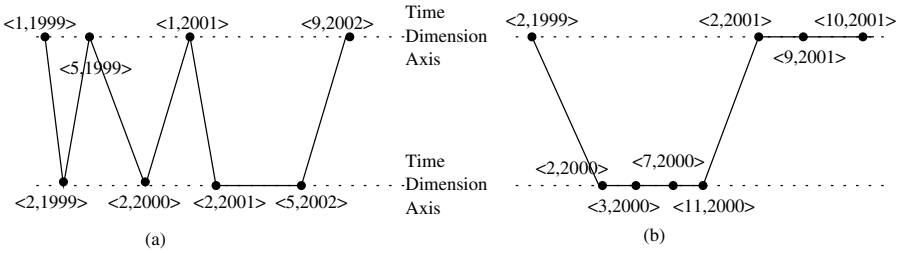


Fig. 4. Time line corresponding to cluster C_1,C_2 and C_3,C_4

Variation-Count: It is described as the no. of lines joining the points (t_i, t_j) where $t_i \in T_1$ and $t_j \in T_2$ obtained from the above process. If the variation-count is more than the threshold then we do not process these clusters further. We have used 6 as a threshold here. The reason is that it is very rare that *author* switches between two *affiliation* more than 6 times. Figure 4(a) describes the time line corresponding to cluster C_1 and C_2 of Table 2 with variation-count 6 and Figure 4(b) shows the time line corresponding to cluster C_3 and C_4 of Table 2 with variation-count 2.

Similarity Computation: Similarity is computed between two clusters if they are non-intersecting in nature or if they are intersecting in nature and variation-count is less than the threshold. Similarity is computed using the 4 attributes namely *co-author*, *co-author affiliation*, *references* and *venue*.

Let $\text{sim}_{cca}(C_i,C_j,A)$ represents the attribute similarity between C_i and C_j. This attribute similarity give the no. of distinct matching attribute value between C_i and C_j. Let th_c, th_{ca} and th_r denote the thresholds for *co-author*, *co-author*

affiliation and *references* respectively. Let $\text{sim}_{cc}(C_i,C_j)$ be the similarity between two clusters C_i and C_j. Formally, for two clusters C_i and C_j; $\text{sim}_{cc}(C_i,C_j) = 1$ if any three of the following four conditions are satisfied.

(1) $\text{sim}_{cca}(C_i,C_j,\text{co-author}) \geq \text{th}_c$ (2) $\text{sim}_{cca}(C_i,C_j,\text{reference}) \geq \text{th}_{ca}$

(3) $\text{sim}_{cca}(C_i,C_j,\text{co-author affl}) \geq \text{th}_r$ (4) $\text{sim}_{cca}(C_i,C_j,\text{venue}) \geq 1$

The possible values of the th_c, th_{ca} and th_r are calculated as below. Here # is used to denote the count operation and *dist* is used to find the distinct values.

$$th_c = \min(\lceil \frac{\#(coauthor_{C_i})}{\#(dist(coauthor_{C_i})) \times \#(C_i)} \rceil, \lceil \frac{\#(coauthor_{C_i})}{\#(dist(coauthor_{C_i})) \times \#(C_j)} \rceil)$$

$$th_c = \min(\lceil \frac{\#(dist(coauthor\ affl_{C_i}))}{\#(C_i)} \rceil, \lceil \frac{\#(dist(coauthor\ affl_{C_j}))}{\#(C_j)} \rceil)$$

$$th_r = \min(|C_i|, |C_j|).$$

If $sim(C_i, C_j) = 1$ then we combine these two clusters otherwise do not combine them. We process the clusters in the ascending order of their rank. We compare the cluster having rank 1 with all other clusters and if the merging criteria is satisfied then we immediately merge those two clusters. The cluster which is merged will never be processed further. After processing first cluster, we process second cluster with the remaining clusters and repeat this process for all the remaining clusters. We repeat this whole process until we get the same clustering result in the two consecutive iterations. This is the *convergence criteria* for *C-C Similarity*.

3.2 R-C-R Similarity Based Clustering

In R-C-R Similarity we compute two types of similarities: record-cluster similarity (R-C Similarity) and record-record similarity (R-R Similarity). The resultant clusters after C-C Similarity are processed next. Here *record* implies all the clusters having single record and *cluster* represents all the clusters having more than one record. First each record is compared with all the clusters and if they can be merged by satisfying a prescribed threshold, we merge them. After this, records are compared with each other. If two records are merged then they form a cluster. Here we use R-R Similarity and R-C Similarity both.

R-C Similarity: For calculating the *R-C Similarity*, we consider 2 attributes-*co-author* and *co-author affiliation* and find their similarities. These attribute similarities provide the no. of distinct matching attribute values from the record and cluster under consideration. Let $\text{sim}_{rc}(r, C)$ represents the R-C Similarity between record r and cluster C while $\text{sim}_{rca}(r, C, A)$ denotes the similarity between attribute A of r and C. Formally, *R-C Similarity* between r and C is calculated by

$$sim_{rc}(r, C) = \frac{\sum_{A \in \mathbf{A}} w_A(\delta t).sim_{rca}(r, C, A)}{\sum_{A \in \mathbf{A}} w_A(\delta t)}$$

Computation of δt: Let $r.t$ is the record(r) time-stamp and $r_i.t$, $i \in [1, n]$ represents the time-stamp of each record of the cluster(C) provided clusters has n records. δt is calculated as follows. $\delta t = \min(|r_1.t - r.t|, |r_2.t - r.t|, \cdots, |r_n.t - r.t|)$

$w_A(\delta t)$ is a function of time and is calculated as $w_A(\delta t) = 1/((\delta t)/12) = 12/(\delta t)$. In case δt is 0, we fix weight to 24 corresponding to each participating attribute.

R-R Similarity: It is computed with the help of 2 attributes that are same as described previously in *R-C Similarity*. Let $\text{sim}_{rra}(r_i, r_j, A)$ represents the similarity between attribute A of two records r_i and r_j and $\text{sim}_{rr}(r_i, r_j)$ expresses the R-R Similarity between r_i and r_j. Formally *R-R Similarity* between two records r_i and r_j is calculated by

$$sim_{rr}(r_i, r_j) = \frac{\sum_{A \in \mathbf{A}} w_A \delta t.sim_{rra}(r_i, r_j, A)}{\sum_{A \in \mathbf{A}} .w_A(\delta t)}$$

and $w_A(\delta t)$ is calculated as follows. $\delta t = |r_i.t - r_j.t|$

4 Evaluation and Analysis

In this section, we describe the different datasets that we have considered in evaluating proposed EMTBD technique. We have also compared our results with DBLP and ArnetMiner. Seven datasets are used for evaluating the performance of our approach where each dataset corresponds to an author. Table 4 shows the different datasets and comparison with 2 existing systems. For evaluation of EMTBD technique, we have used three basic measures -*precision*, *recall* and *f-measure* as found in [10]. Here, we have used 0.8 as threshold values in R-C and R-R similarity.

For *Xin Dong* dataset, we could not find all the records in ArnetMiner. In case of *Bin Yu* dataset, the *initials of email-id* is providing sufficient evidences for merging two clusters. In intersecting nature of clusters the *initials of email-id* is same within the same *affiliation*. While evaluating the performance of *Jim*

Table 4. Statistics about the datasets and comparison of performance. NR: no. of records, NE: no. of entities, NER: No. of records per entity.

Name	NR*	NE*	f-measure			NER*
			DBLP	Arnetminer	EMTBD	
Xin Dong	55	10	88.63	NA	97.37	{38,1,1,4,1,1,6,1,1,1}
Ajay Gupta	26	9	50.83	60.00	97.95	{2,12,1,4,2,1,1,2,1,}
Samrat Goswami	17	1	100	100	93.75	{17}
Bin Yu	124	21	34.10	52.70	100.00	{ 50,22,11,7,7,2,2,1,2, 2,3,1,1,4,1,1,1,1,2,1,2}
M. Hasanuzzaman	16	3	62.85	62.85	94.23	{5,1,10}
Jim Smith	38	5	55.94	45.80	100.00	{17,17,2,1,1}
David Jensen	52	4	91.97	86.75	97.87	{48,2,1,1}

Smith dataset, the attribute *references* provides the strong evidence. In this dataset 33% of the *references* between two clusters are matching. We are getting low recall (recall = 88.23 while f-measure = 93.75) for dataset *Samrat Goswami*. The records in this dataset belongs to single entity but it has been split into two entities- one having 16 records and another having 1 record. The reason is that nearly at the same time (1 month difference) two papers are published without having any common attribute value.

5 Conclusion and Future Scope

The proposed technique is specifically applicable for bibliographic database. Both temporal and non-temporal attributes are considered for finding the links between different records. The technique is based on similarity check and clustering of records. The thresholds that are used for similarity computation are computed dynamically and depend on the number of records available. All these concepts have made the proposed EMTBD a very sound one. This is well supported by the given comparative results with DBLP and ArnetMiner. This clearly shows that the proposed technique outperforms the other two techniques in many cases. In future, we like to generalize this approach so that it can be applicable on other type of databases also.

References

1. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD, pp. 39–48 (August 2003)
2. Chaudhuri, S., Chen, B.-C., Ganti, V., Kaushik, R.: Example-driven design of efficient record matching queries. In: VLDB, pp. 327–338 (September 2007)
3. Fan, W., Jia, X., Li, J., Ma, S.: Reasoning about record matching rules. VLDB 2(1), 407–418 (2009)
4. Gal, A., Atluri, V.: An authorization model for temporal data. In: CCS, pp. 144–153 (November 2000)
5. Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: SIGMOD, pp. 127–138 (May 1995)
6. Li, P., Dong, X.L., Maurino, A., Srivastava, D.: Linking temporal records. VLDB 4(11), 956–967 (2011)
7. Li, P., Wang, H., Tziviskou, C., Dong, X.L., Liu, X., Muarino, A., Srivastava, D.: CHRONOS: facilitating history discovery by linking temporal records. VLDB 5(12), 2006–2009 (2012)
8. Li, S., Cong, G., Miao, C.: Author name disambiguation using a new categorical distribution similarity. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part I. LNCS, vol. 7523, pp. 569–584. Springer, Heidelberg (2012)
9. Wang, J., Li, G., Yu, J.X., Feng, J.: Entity Matching: how similar is similar. VLDB 4(10), 622–633 (2011)
10. Yin, X., Han, J., Yu, P.S.: Object Distinction: distinguishing objects with identical names. In: ICDE, pp. 1242–1246 (April 2007)

On Efficient Map-Matching According to Intersections You Pass By

Yaguang Li[1,3], Chengfei Liu[2], Kuien Liu[1], Jiajie Xu[1], Fengcheng He[1,3], and Zhiming Ding[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{yaguang,kuien,jiajie,fengcheng,zhiming}@nfs.iscas.ac.cn
[2] FICT, Swinburne University of Technology, Australia
cliu@swin.edu.au
[3] University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. Map-matching is a hot research topic as it is essential for Moving Object Database and Intelligent Transport Systems. However, existing map-matching techniques cannot satisfy the increasing requirement of applications with massive trajectory data, e.g., traffic flow analysis and route planning. To handle this problem, we propose an efficient map-matching algorithm called *Passby*. Instead of matching every single GPS point, we concentrate on those close to intersections and avoid the computation of map-matching on intermediate GPS points. Meanwhile, this efficient method also increases the uncertainty for determining the real route of the moving object due to less availability of trajectory information. To provide accurate matching results in ambiguous situations, e.g., road intersections and parallel paths, we further propose *Passby**. It is based on the multi-hypothesis technique and manages to maintain a small but complete set of possible solutions and eventually choose the one with the highest probability. The results of experiments performed on real datasets demonstrate that Passby* is efficient while maintaining the high accuracy.

Keywords: Efficient Map-matching, Multi-Hypothesis Technique.

1 Introduction

The past few years have seen a dramatic increase of GPS-embedded and handheld navigation devices. These devices allow to record accurately the spatial displacement of moving objects. Given their low set-up cost, large volumes of data can be easily generated. In order to provide a range of Intelligent Transport Systems services, e.g., traffic flow analysis [1], route planning [2], hot route finder [3], we need to analyze large amounts of trajectory data. An essential pre-processing step that matches the trajectories to the road network is needed. This technique is commonly referred as map-matching.

However, existing map-matching techniques cannot fully satisfy the increasing requirement of applications with massive trajectory data. Some local/increment map-matching methods [4,5] are fast but generate inconsistent matching results

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 42–56, 2013.

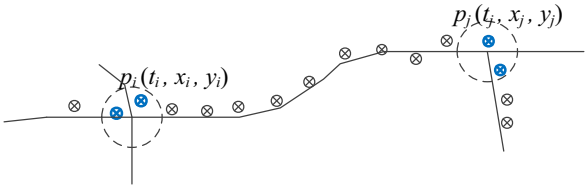


Fig. 1. An example of Passby

since they ignore the correlation between subsequent points and topological information of the road network. Meanwhile, more advanced map-matching methods, e.g., global approach [6,7] and statistical approach [8,9], manage to be quite accurate, but suffer from high computational complexity.

To handle this problem, we propose an efficient map-matching algorithm called Passby. Instead of matching every single GPS point, we concentrate on those close to intersections and avoid the computation of map-matching on intermediate GPS points. In the circumstances of constrained road network, if a moving object passes by the start vertex of edge e at t_i, and the end vertex of e at a reasonable time t_j, the object is then supposed to be on e at the time interval between t_i and t_j. Figure 1 illustrates an example of such case, i.e., we just have to match these thick blue points close to intersections instead of analyzing every single point. While cutting the cost of computing on intermediate GPS points makes map-matching more efficient, nevertheless it is not at no cost. The uncertainty for determining the real route of the moving object also arises due to less availability of trajectory information. The challenge of this work is to keep providing accurate matching results in ambiguous situations, e.g., road intersections and parallel paths. To handle this challenge, we propose another algorithm called Passby*, which is based on the multi-hypothesis technique, to maintain all the possible solutions in situations of ambiguity and eventually choose the one with the highest probability. Meanwhile, several efficient approaches for hypothesis management are developed to further improve the performance of Passby*.

To summarize, this paper makes the following contributions:

- We propose two efficient map-matching algorithms, Passby and Passby*. They avoid large parts of computation by only performing map-matching on GPS points close to intersections and inferring the remaining ones.
- We develop a set of novel approaches for hypothesis creation, evaluation and management which make the algorithm both efficient and accurate.
- We present a solution to edge simplification and edge measurement error. As Passby* mainly bases on positions of intersections and road connectivity, it is robust when geometric information of the edge is not available or not accurate enough.
- We conduct experimental study on real dataset and demonstrate the efficiency and accuracy of the proposed algorithms.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 gives the problem statement. The proposed algorithm is described in

Section 4. The results of experiments and the analysis are presented in Section 5. Finally, conclusions and future work are summarized.

2 Related Work

In this section, we present a brief overview of previous studies on map-matching and the multi-hypothesis technique.

2.1 Map-Matching Techniques

Map-matching is commonly referred as the process of aligning a sequence of observed positions with the road network. Approaches for map-matching algorithms can be categorized into three groups [6]: local/incremental method [4,5,10], global method [6,7,11], and statistical method [8,9]. The local/incremental method tries to find local match of geometries, and performs matching based on the previous matching result of a point. The global method aims to find a global optimal match for the entire trajectory and a path in the road network. The algorithm in [11] is based on Frechét distance and its variant. [6] proposes an algorithm that takes temporal information into consideration and manages to get a high accuracy for low-sampling-rate GPS trajectories. Statistical method is also widely used. A method based on the Bayesian classifier is presented in [9]. [8] proposes an algorithm that takes advantage of the rich information extracted from the historical trajectories to infer the real routes.

Many advanced techniques are also integrated with map-matching [12]. [13] proposes a fuzzy logical based method that is able to take noisy, imprecise input and yield crisp output. [7] proposes a method based on the Hidden Markov Model which can deal with the inaccuracies of the location measurement systems.

Most existing map-matching algorithms have to perform map-matching on every single trajectory point which is not always necessary. Instead of matching every single GPS point, we concentrate on those close to intersections and infer the results for the remaining points to avoid extra computation.

2.2 Multi-Hypothesis Technique

The Multi-Hypothesis Technique(MHT) is a method to track multiple targets under the clutter environment using a likelihood function [14]. To realize a map matching method using the MHT, pseudo-measurements are generated utilizing adjacent roads of GPS position and the MHT is reformulated as a single target problem [15]. The main advantage of multi-hypothesis map-matching over a mono-hypothesis approach is that it maintains all the possible solutions in situations of ambiguity and eventually chooses the one with the highest probability. Two map-matching methods using the MHT are proposed in [15] and [16]. These previous works are more focused on the accuracy than the efficiency which makes them unsuitable for processing massive trajectory data. Moreover, these methods have to use extra information from the dead reckoning device, e.g., heading, acceleration and speed, which is not always available.

3 Problem Statement

We have a standard road network representation in which intersections are represented by vertices and roads are represented by directed edges and the geometric detail of roads are described by polylines.

Definition 1 (Road Network). *The road network G is defined as:*

$$G = (V, E) \tag{1}$$

where V is the set of vertices while E is the set of directed edges. A vertex of G is defined as: $v = (vid, lat, lng)$, where vid is the identifier of v while lat and lng denote the latitude and longitude of v. An edge of G is defined as: $e = (eid, geo, len, v_{from}, v_{to})$, where eid is the identifier of e, len is the length of e, v_{from} and v_{to} represent the start and the end vertices of e, geo is a polyline representing the geometry of e.

In the most simplified road network described in [17], *geo* is not available which will pose a great challenge to map-matching algorithms. For robustness evaluation, we will also conduct experiments on the most simplified road network.

Definition 2 (Path). *A path P is defined as a sequence of edges:*

$$P = (e_i)_{i=1}^{n} \; n \geq 1 \; \wedge \; \forall 1 \leq i < n \;\; e_{i+1}.start = e_i.end \tag{2}$$

The start vertex and the end vertex of P are defined as:

$$P.start = e_1.start \qquad P.end = e_n.end$$

Definition 3 (Trajectory). *The trajectory of a moving object is defined as a sequence of points with timestamps:*

$$T = (p_i)_{i=1}^{n} = (t_i, lat_i, lng_i)_{i=1}^{n}$$

We use $T[i]$ to represent the ith point of T and $\overrightarrow{p_i, p_{i+1}}$ to represent the ith trajectory segment of T. $\overrightarrow{p_i, p_{i+1}}$ is defined as the directed line segment formed by p_i and p_{i+1}.

Definition 4 (Matching Result). *Given a trajectory $T = (p_i)_{i=1}^{n}$, the matching result R is defined as:*

$$R = (p_i, e_i)_{i=1}^{n} \tag{3}$$

where e_i denotes the edge that p_i be matched to.

Definition 5 (Matching Accuracy). *Given the matching result $R = (p_i, e_i)_{i=1}^{n}$ and the ground truth $R_t = (p_i, \hat{e}_i)_{i=1}^{n}$, the accuracy A is defined as:*

$$A = \frac{\sum_{i=1}^{n} \theta_i}{n} \qquad \theta_i = \begin{cases} 1 & \text{if } R.e_i = R_t.\hat{e}_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Problem Statement The **Map-matching** problem can be described as, given a road network G and a trajectory T, how to efficiently find a matching result R, with the goal of maximizing the matching accuracy A.

4 Proposed Algorithms

In this section, we first illustrate the key observations of Passby followed by the algorithm description and implementation, then we describe several novel approaches for the multi-hypothesis model in detail including hypothesis creation, evaluation and management.

Observation 1. *In most cases, if a moving object passes by both the start and the end intersections of an edge within a reasonable period of time, the object is supposed to be on that edge during the time interval.*

According to Observation 1, we can improve the efficiency of map-matching algorithm by avoiding the computation on intermediate GPS points.

Observation 2. *The representation of vertices is generally more accurate than edges especially when edges are simplified.*

The most commonly used representation of vertex and edge in a road network is point and polyline. Sometimes, the road network is simplified [17,18], the deviations between real edges and polylines can be up to hundreds of meters due to edge simplification or edge measurement error while the representation of vertices tends to be more accurate.

4.1 Passby: A Baseline Approach

Based on above observations, we design a map-matching algorithm called Passby, which mainly depends on positions of intersections and road connectivity. In particular, this algorithm does not need either the heading information or the instantaneous speed of the vehicle from the dead reckoning device.

One of the key processes in Passby is to estimate how likely the moving object passes by a certain vertex. We measure it using the passby probability.

Definition 6 (Passby Probability). *The passby probability is defined as the likelihood that the moving object passes by the vertex v.*

$$F_p(T, v) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(\lambda-\mu)^2}{2\sigma^2}} \qquad \lambda = d(T, v) = \min_{0<i<|T|-1} d(\overrightarrow{p_i, p_{i+1}}, v)$$

where λ is the distance between the trajectory T and the vertex v, i.e., the minimum distance between the segments of T and v. We use a zero-mean normal distribution with a standard deviation of 6.4m for the training dataset.

Algorithm 1 gives an overview of Passby. The function **init**(G, T, i) takes three arguments: the road network G, the trajectory T and the start index i, to find the first passed vertex v_c that $d(T, v_c)$ is less than a threshold σ_p, e.g., 25m, and the index of corresponding trajectory segment i_c. This can be efficiently calculated by building indices on GPS points and vertices. The function **pass**(T, v, i) takes three arguments: the trajectory T, the vertex v, and start index of trajectory segment i, and returns the index of the nearest trajectory segment from v, or 0 if no valid trajectory segment is found.

Algorithm 1. Passby(G, T)

Input:Road network G, trajectory $T = (p_i)_{i=1}^{n} = (t_i, lat_i, lng_i)_{i=1}^{n}$

1: $(v_c, i_c) \leftarrow$ **init**$(G, T, 1)$
2: **while** $i_c \leq n$ **do**
3: $S_c \leftarrow \emptyset,\ S_e \leftarrow \{e | e.start = v_c\}$
4: **for each** $e \in S_e$ **do**
5: $i \leftarrow$ **pass**$(T, e.end, i_c)$
6: **if** $i > 0$ **then**
7: $S_c \leftarrow S_c \cup \{(i, e)\}$
8: **if** $S_c \neq \emptyset$ **then**
9: $t \leftarrow \arg\min_{t \in S_c}(t.i)$
10: match $\{p_{i_c+1}, \ldots, p_{t.i}\}$ to edge $t.e$
11: $i_c \leftarrow t.i,\ v_c \leftarrow t.e.end$
12: **else**
13: $(v_c, i'_c) \leftarrow$ **init**(G, T, i_c)
14: match $\{p_{i_c+1}, \ldots, p_{i'_c}\}$ to nearest edges
15: $i_c \leftarrow i'_c$

First, the algorithm performs the initialization to get the first passed vertex v_c and the corresponding index of the trajectory segment i_c. (line 1). Then the algorithm repeats the following process until the end of trajectory: 1) it retrieves all the candidate edges (line 3); 2) it filters out infeasible edges using passby probability (line 4 - line 7); 3) if valid edges exist, the edge $t.e$ with the smallest index is selected (line 8 - line 9), then the algorithm matches the points between p_{i_c+1} and $p_{t.i}$ to $t.e$ (line 10), otherwise a re-initialization will be made to find the next vertex v_c and corresponding index (line 13). The intermediate points will be matched to their nearest edges (line 14). As illustrated in Figure 1, the algorithm needs only to match the initial and the final points in most cases and is consequently very efficient.

4.2 Passby*: A Multi-Hypothesis Based Approach

The Passby algorithm works well in the premise that every passed vertices can be found and the one closest to the moving object is the real passed vertex. However, this assumption may not always hold, e.g., several vertices can be quite close to the moving object at the same time while the real passed vertex might not be close enough to be found. To generate stable matching results in ambiguous situations, we propose another algorithm called Passby*, which is integrated with the multi-hypothesis technique to maintain all the possible solutions in situations of ambiguity and eventually choose the one with the highest probability.

4.2.1 Hypothesis Creation

The traditional hypothesis creation method usually generates new hypotheses at each step, which will result in an exponential computational complexity. To be efficient, we redesigned the rule of hypothesis creation.

Algorithm 2. createCands(G, T, sp)

Input: Road network G, trajectory $T = (t_i, lat_i, lng_i)_{i=1}^{n}$, path structure sp
Output: A set of candidate path structures: S_{sp}

1: $S_{sp} \leftarrow \emptyset$
2: $S_v \leftarrow \{v | d(v, T[sp.i]) + d(v, T[sp.i+1]) \leq 2a\}$
3: $S_P \leftarrow$ **buildPath**(G, S_v, sp)
4: **for each** $P \in S_P$ **do**
5: $\quad i \leftarrow$ **pass**$(T, P.end, sp.i)$
6: $\quad$ **if** $i > 0$ **then**
7: $\quad\quad f \leftarrow sp.f \cdot F_p(T, P.end) \cdot F_t(T, P)$
8: $\quad\quad S_{sp} \leftarrow S_{sp} \cup \{(P.end, i, f, sp\}$
9: **return** S_{sp}

Algorithm 2 shows the pseudo-code of candidate creation process. Function **createCands**(G, T, sp) takes three arguments: the road network G, the trajectory T, and the path information structure $sp : (v, i, f, ptr)$ which contains information about the candidate path, where v is the vertex, i is the index of the trajectory segment that is nearest to the v, f is the score of the path and ptr is the pointer to the parent structure, which is used to get the whole path.

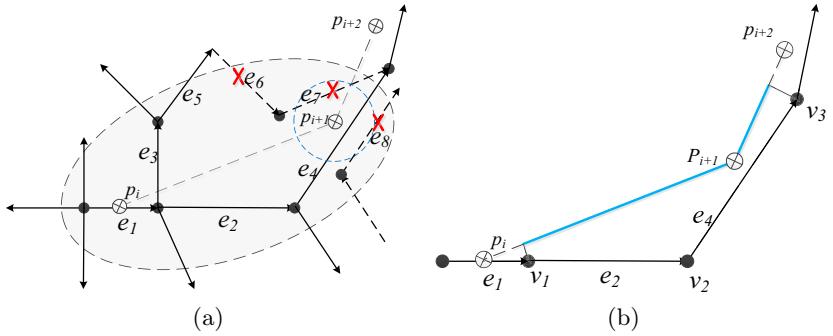


Fig. 2. Examples of hypothesis creation

Take Figure 2(a) for example, the algorithm first retrieves all the edges whose start vertices lie within the error ellipse that has a major axis of $2a$ (line 2). e_6 is not a valid edge as its start vertex falls outside the ellipse. e_7, e_8 won't be considered either as they are not connected with valid edges. Then the algorithm builds candidate paths in line 3. These paths start from $sp.v$ and are extended with candidate edges based on road connectivity. Finally, the algorithm finds these candidate paths whose end vertices are considered passed by the moving object, and calculates their scores using passby probability and vertex transition probability (line 4 - line 8).

Definition 7 (Vertex Transition Probability). *The vertex transition probability is used to measure the likelihood that the moving object transfers between vertices, and is defined as:*

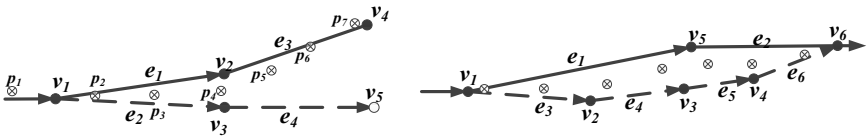
$$F_t(T,P) = \exp(-\alpha_t \frac{\sum_{e \in P} \text{len}(e)}{\sum_{e \in P} \text{len}(\text{proj}(e,T))}) \quad (5)$$

where α_t is the scaling factor, $\text{len}(e)$ returns the length of e, and $\text{proj}(e,T)$ returns the projection of e on T. Figure 2(b) shows the candidate path $\{e_2, e_4\}$, the thick blue line represents the projection of the path on T. The efficiency of this hypothesis creation method will be discussed in Section 4.2.3.

4.2.2 Solutions to Ambiguous Situations

In practice, there exist many ambiguous situations, e.g., Y-junctions [12] and parallel paths, which pose a challenge to map-matching.

Definition 8 (Ambiguous Situation). *An ambiguous situation is defined as the situation in which there exists a path P in the candidate set, the end vertices of more than one of P's candidate edges are close to the trajectory.*



(a) Y-juntions (b) Parallel paths

Fig. 3. Examples of ambiguous situations

Figure 3(a) shows a case of Y-junctions problem with vertices represented by solid circles are considered possibly passed by the moving object while empty ones not, e.g., v_5. It is difficult to determine whether p_2, p_3, p_4 should be matched to e_1 or to e_2. [11] describes a solution by performing certain steps of look-ahead. However, this method is quite time-consuming especially when the number of look-ahead is large [10].

Observation 3. *In most cases, the deviation between the wrong path and the real route of the moving object tends to increase with the elapse of time.*

This simple observation is helpful to solve the Y-junctions problem. We use the example in Figure 3(a) to explain. Let P be the path to be extended, as both v_2 and v_3 satisfy the criteria, two new paths $P_1 = P \cup \{e_1\}$ and $P_2 = P \cup \{e_2\}$ will be created. As stated in Observation 3, the deviation between the real route and the wrong path P_2 increases, there is no GPS points near the candidate edges of P_2, thus it will not be extended any more, i.e., this hypothesis will be automatically discarded. However, Observation 3 will not hold in another kind of ambiguous situation called Parallel Paths.

Definition 9 (Parallel Paths). *Parallel Paths is defined as a set of paths S_{pp} satisfying the following two conditions:*

- *The maximum Hausdorff distance [19] between any two paths in S_P is lower than $2\sigma_p$.*

$$\max_{P_i, P_j \in S_{pp}} h(P_i, P_j) < 2\sigma_p$$

- *All of these paths have identical start and end vertices.*

$$\forall P_i, P_j \in S_{pp} \quad P_i.start = P_j.start \wedge P_i.end = P_j.end$$

Figure 3(b) gives an example of parallel paths. Both of these two paths start from v_1 and end at v_6. As these two paths come quite close to each other, all the vertices in the two paths can be possibly passed by the moving object, which makes it difficult to find the real path. After analyzing the relation between the cost of a path, i.e., the minimum time required to pass it, and the actual time used to go through it, we found Observation 4.

Observation 4. *The cost of the real path tends to be similar to the actual time used to go through it.*

Although the lengths of the parallel paths might be nearly the same, the differences between their costs are relatively large, e.g., the highway and the service road nearby. Therefore, we are likely to find the real path by further considering the cost similarity.

Definition 10 (Cost Similarity). *Given a path P and a trajectory T, the cost similarity $F_c(T, P)$ is defined as follows:*

$$F_c(T, P) = \prod_{e \in P} \exp(-\alpha_c |c_e - \hat{c}_e|) = \exp(\sum_{e \in P} -\alpha_c |c_e - \hat{c}_e|) \tag{6}$$

$$c_e = T[i].t - T[j].t \qquad \hat{c}_e = \frac{\text{len}(e)}{s_e}$$

where c_e is the actual time used to pass e, $\hat{c}_e$ is the cost the e, α_c is the scaling factor for cost similarity. i and j are the indices of trajectory segment that pass the start vertex and the end vertex of e, $\text{len}(e)$ is the length of e, s_e is the speed limitation of e.

4.2.3 Management of Candidate Paths

The candidate management is mainly used to reduce the candidate size while preserving all possible solutions. It consists of two aspects: pruning and confirmation. Pruning is the process of eliminating infeasible candidate paths while confirmation is the process of confirming a candidate path as the real path.

Pruning happens in the following conditions: 1)The ratio of the score of a hypothesis to the largest score is lower than a threshold σ_A, which usually happens at Y-junctions. 2) In the case of parallel paths, all the parallel paths should be merged into a single one.

Confirmation happens in the following conditions: 1) The ratio of the largest score to the next largest one exceeds a threshold σ_B. 2) There exists only one hypothesis. σ_A and σ_B are two thresholds used to control the candidate size.

Next, we will show that the number of hypotheses will only increases in ambiguous situations. In order to increase the number of hypotheses, at least two sub-paths must be created from a path. Besides, one of the prerequisites of hypothesis creation is that the end vertex of the path must be close enough to the trajectory, i.e., the ambiguous situation.

4.2.4 Overview of Passby*

Figure 4 gives a description of the procedure of the algorithm. Figure 4(a) shows the road network and the GPS points, while Figure 4(b) demonstrates the searching process. As mentioned before, vertices represented by solid circles in figures are considered possibly passed by the moving object while empty ones not. The red cross denotes a clip of the search graph, with corresponding path being removed from the candidate set.

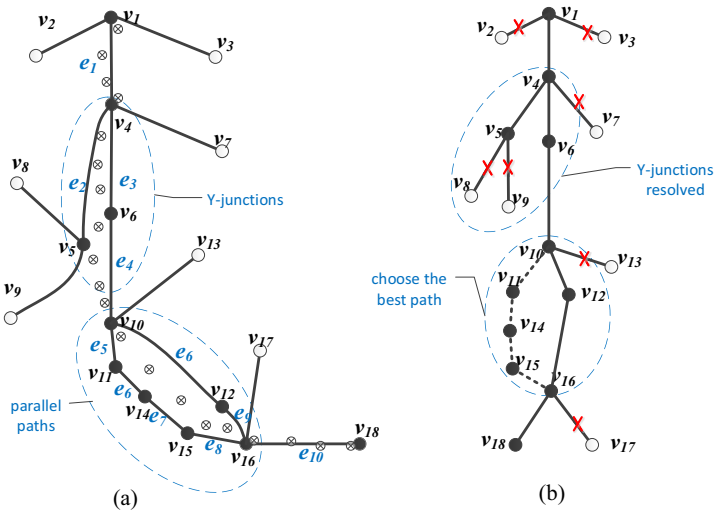


Fig. 4. An example of the Passby* algorithm

Algorithm 3 outlines the framework of Passby*. S_O, S_P, S_{pp} are all sets of path information structures, i.e., $sp : (v, i, f, ptr)$. Function **getInitialCands**(G, T) is used to get the initial vertices and evaluate them with passby probability. Function **createCands**(G, T, sp) is used to get candidate paths with the method proposed in Section 4.2.1. Function **getBestPath**(S_{pp}, T) is used to get the set of parallel paths and further evaluate them with cost similarity to get the best path.

Take Figure 4(a) for example, the algorithm first gets initial paths in line 4, and then pop the first item from S_O, i.e., the item with smallest i, as sp_c (line 5). Hypothesis creation method proposed in Section 4.2.1 are used to generate candidate paths (line 6). These candidate paths are processed in line 7 to line 13. Line 8 indicates the case of parallel paths, and the best candidate is selected using

Algorithm 3. Passby* (G, T)

Input:Road network G, trajectory $T = (t_i, lat_i, lng_i)_{i=1}^{n}$
Output:The matching result: R

1: $S_O \leftarrow \emptyset$; $sp_c \leftarrow (null, 1, 1, null)$ //(v, i, f, ptr)
2: **while** $sp_c.i \leq n$ **do**
3: **if** $S_O = \emptyset$ **then**
4: $S_O \leftarrow$ **getInitialCands**$(G, T, sp_c.i)$
5: **while** $S_O \neq \emptyset \wedge sp_c.i \leq n$ **do**
6: $sp_c \leftarrow$ **first**(S_O); $S_O \leftarrow S_O \backslash \{sp_c\}$;
7: $S_P \leftarrow$ **createCands**(G, T, sp_c) //Section 4.2.1
8: **for each** sp in S_P **do**
9: $S_{pp} \leftarrow \{sp' | sp' \in S_O \wedge sp'.i = sp.i \wedge sp'v = sp.v\}$
10: **if** $S_{pp} \neq \emptyset$ **then**
11: $sp_{best} \leftarrow$ **getBestPath**(S_{pp}, T) //Section 4.2.2
12: $S_O \leftarrow S_O \backslash S_{pp} \cup \{sp_{best}\}$
13: **else**
14: $S_O \leftarrow S_O \cup \{sp\}$
15: $S_O \leftarrow$ **pruneConfirm**(S_O) //Section 4.2.3
16: $sp_c \leftarrow \arg\max_{sp \in S_O}(sp.f)$
17: $R \leftarrow$ **getMatchResult**(sp_c)
18: **return** R

method in Section 4.2.2 (line 10). The pruning and confirmation are performed in line 14. Finally, the algorithm finds the path with the maximum score in S_O (line 15), i.e., $\{e_1, e_3, e_4, e_6, e_9, e_{10}\}$, and generates the matching result (line 16).

4.3 Theoretical Analysis

Now we will analyze the complexity of the **Passby*** algorithm. Let n be the length of the trajectory T, l be the maximum number of edges in the error ellipse used in the hypothesis generation process, and k be the maximum number of candidates.

The time complexity of the function **createCands**, **pruneConform** and **getMatchResult** are $O(l \log(l))$, $O(k)$ and $O(n)$. In the worst-case, hypothesis creation might be performed on every single trajectory point, so the Passby* algorithm has the time complexity of $O(nkl \log(l) + nk + n) = O(nkl \log(l))$. In practice, k is usually quite small, thus the time complexity of Passby* is close to $O(nl \log(l))$. Furthermore, as indicated by Figure 1, large part of computation is avoid, so the constant factor is actually quite small, this is also confirmed by the experiment.

5 Experiments

In this section, we first present the experimental settings, then we evaluate the efficiency and accuracy of the proposed algorithms, finally we show their performances on the most simplified road network.

5.1 Dataset and Experimental Setup

In our experiment, we use the road network and trajectory data provided by ACM SIGSPATIAL Cup 2012 [20] , which is a GIS-focused algorithm competition hosted by ACM SIGSPATIAL. The road network graph contains 535,452 vertices and 1,283,540 road segments. Ground truth files are included in trajectory data file, which are used in results verification. These algorithms have been implemented with C++ on Visual Studio express platform, the results of the experiments are taken on a computer with Intel Core2 Dual CPU T8100 2.1 GHz and 3 GB RAM.

Baseline Algorithms. We compare proposed algorithms with the incremental algorithm: IMM05 [11], and the Hidden Markov Model based algorithm: NK09 [7]. IMM05 performs matching based on three aspects: the distance, the direction and a point's previous matching result, and is known to have a low time complexity. NK09 is well-known for its high accuracy and the ability to deal with noise.

For IMM05, we use the suggested settings in [11]: $\mu_d = 10, \alpha = 0.17, n_d = 1.4, \mu_a = 10$, $n_\alpha = 4$. For NK09, we use the following assignment : $\sigma = 4.1$, and $\beta = 5$. Preprocessing and optimization suggested in [7] are made to NK09. To further speed up the algorithm, roads that are more than 50 meters away from the GPS point are eliminated from the candidate set. For Passby and Passby*, we use : $\sigma = 6.4, \sigma_p = 25, \alpha_t = 65, \alpha_c = 4, \sigma_A = 0.1, \sigma_B = 3$.

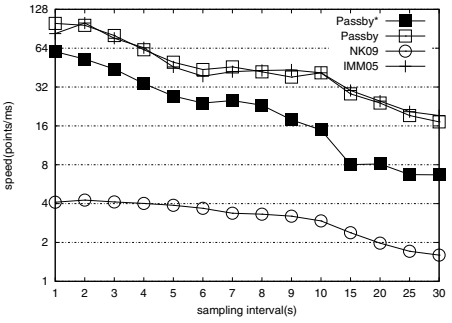


Fig. 5. Efficiency

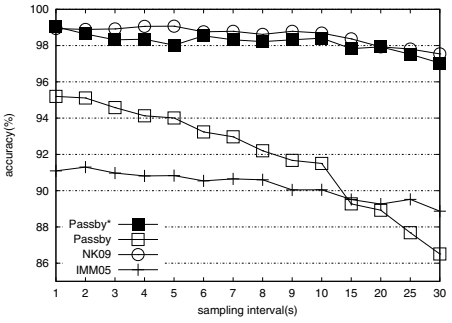


Fig. 6. Accuracy

5.2 Matching Efficiency with Different Sampling Interval

As shown in Figure 5, Passby* is 4∼10 times faster than the NK09, and is even comparable to IMM05 when sampling interval is small. In addition, the matching speed of Passby is a little faster than IMM05 . With the increase of the sampling interval, the speeds of Passby and Passby* begin to decrease due to the fact that less computation can be avoided.

The following reasons may contribute to the high efficiency of Passby*:

- Computation of map-matching on a large part of GPS points is avoided. In most cases, Passby* needs only to match the initial and the final points. Moreover, there is less shortest path calculation in Passby* than NK09 which is quite time-consuming.
- The refined hypothesis management method is efficient. With the hypothesis creation method proposed in Section 4.2.1, the number of hypotheses is usually quite small.

5.3 Matching Accuracy with Different Sampling Interval

As shown in Figure 6, the matching accuracy of Passby* is quite comparable to NK09, and much higher than IMM05. Meanwhile, the accuracies of all these algorithms decrease with the increase of sampling interval as the deviation between the real route of the moving object and the trajectory represented by polyline becomes larger.

In particular, the matching accuracy of Passby decreases significantly with the increase of sampling interval. This is mainly because Passby simply choose the nearest vertex as the real passed one which is less accurate when the distances between the trajectory and passed vertices become larger. In contrast, Passby* maintains all the possible paths, and eventually chooses the best one which makes it less sensitive to the increase of sampling interval. Several reasons may contribute to the high accuracy of Passby*:

- Passby* evaluates candidate path in terms of both temporal and spatial factors, e.g, passby probability, vertex transition probability and cost similarity.
- The refined multi-hypothesis model is used to effectively maintain a set of candidate paths and eventually choose the best one.
- Passby* is less dependent on the geometric detail of the edge, e.g., it needs no projection to the edge, which makes it robust to edge measurement errors.

5.4 Matching Accuracy on the Most Simplified Road Network

In the most simplified road network, as mentioned in Definition 1, the edge detail, i.e., *geo*, is omitted. This will significantly reduce the size of the digital map. However, the distance between the trajectory point and the simplified edge becomes much larger, and it sometimes can be up to hundreds of meters. Figure 7 gives an example of edge simplification. To evaluate the robustness of the algorithms to edge simplification, we test the algorithms on the most simplified road network.

Figure 8 illustrates the matching accuracies on both the original and the most simplified road network when the sampling interval is 10s. The accuracy of Passby* merely changes while the accuracies of other two algorithms suffer from significant decreases. This is because Passby* matches the trajectory mainly based on intersections and is less dependent on edge detail information.

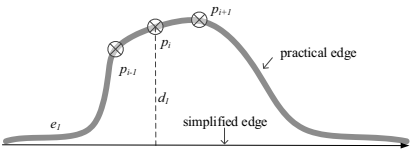


Fig. 7. Edge simplification

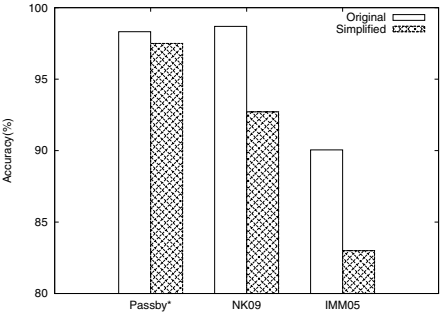


Fig. 8. Matching accuracy on the most simplified road network

6 Conclusion and Future Work

In this paper, we investigate the problem of how to improve the efficiency of map-matching, and propose two novel algorithms called Passby and Passby*. They perform map-matching according to intersections that the moving object passes by. Taking advantage of the proposed candidate generation and evaluation methods, Passby* manages to efficiently provide accurate matching results in ambiguous situations. In addition, Passby* is less dependent on the edge detail information, and is robust to edge measurement error and edge simplification. We conduct experimental study on real dataset to evaluate the performance of proposed algorithms. The results of the experiments indicate that Passby* is both efficient and accurate.

In the future work, we plan to further improve the algorithm to better process GPS data of low sampling rate.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (Nos. 61202064, 91124001), the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06020600), and the Key Research Program of the Chinese Academy of Sciences (NO. KGZD-EW-102-3-3).

References

1. Liu, K., Deng, K., Ding, Z., Li, M., Zhou, X.: Moir/mt: Monitoring large-scale road network traffic in real-time. PVLDB 2(2), 1538–1541 (2009)
2. Gonzalez, H., Han, J., Li, X., Myslinska, M., Sondag, J.P.: Adaptive fastest path computation on a road network: a traffic mining approach. In: VLDB, pp. 794–805 (2007)
3. Li, X., Han, J., Lee, J.-G., Gonzalez, H.: Traffic density-based discovery of hot routes in road networks. In: Papadias, D., Zhang, D., Kollios, G. (eds.) SSTD 2007. LNCS, vol. 4605, pp. 441–459. Springer, Heidelberg (2007)

4. White, C.E., Bernstein, D., Kornhauser, A.L.: Some map matching algorithms for personal navigation assistants. Transportation Research Part C: Emerging Technologies 8(1-6), 91–108 (2000)
5. Greenfeld, J.S.: Matching gps observations to locations on a digital map. In: Transportation Research Board. Meeting, Washington, D.C (2002)
6. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate gps trajectories. In: GIS, Seattle, Washington, pp. 352–361 (2009)
7. Newson, P., Krumm, J.: Hidden markov map matching through noise and sparseness. In: GIS, Seattle, WA, USA, pp. 336–343 (2009)
8. Zheng, K., Zheng, Y., Xie, X., Zhou, X.: Reducing uncertainty of low-sampling-rate trajectories. In: ICDE, Washington, DC, USA, pp. 1144–1155 (2012)
9. Pink, O., Hummel, B.: A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In: ITSC, pp. 862–867. IEEE (2008)
10. Wenk, C., Salas, R., Pfoser, D.: Addressing the need for map-matching speed: Localizing globalb curve-matching algorithms. In: SSDBM, Washington, DC, USA, pp. 379–388 (2006)
11. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: VLDB, Trondheim, Norway, pp. 853–864 (2005)
12. Quddus, M.A., Ochieng, W.Y., Noland, R.B.: Current map-matching algorithms for transport applications: State-of-the art and future research directions. Transportation Research Part C: Emerging Technologies 15(5), 312–328 (2007)
13. Syed, S., Cannon, M.: Fuzzy logic based-map matching algorithm for vehicle navigation system in urban canyons. In: National Technical Meeting of The Institute of Navigation, San Diego, CA, pp. 982–993 (2004)
14. Reid, D.: An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control 24(6), 843–854 (1979)
15. Pyo, J.S., Shin, D.H., Sung, T.K.: Development of a map matching method using the multiple hypothesis technique. In: Proceedings of 2001 Intelligent Transportation Systems, pp. 23–27. IEEE, Oakland (2001)
16. Abdallah, F., Nassreddine, G., Denoeux, T.: A multiple-hypothesis map-matching method suitable for weighted and box-shaped state estimation for localization. IEEE Transactions on Intelligent Transportation Systems 12(4), 1495–1510 (2011)
17. Liu, K., Li, Y., He, F., Xu, J., Ding, Z.: Effective map-matching on the most simplified road network. In: GIS, Redondo Beach, CA, USA, pp. 609–612 (2012)
18. Zhou, J., Golledge, R.: A three-step general map matching method in the gis environment: travel/transportation study perspective. International Journal of Geographical Information System 8(3), 243–260 (2006)
19. Alt, H., Guibas, L.: Discrete geometric shapes: Matching, interpolation, and approximation. In: Handbook of Computational Geometry, Amsterdam, pp. 121–153 (1999)
20. ACM SIGSPATIAL Cup 2012: Training data sets (2012), `http://depts.washington.edu/giscup/home`

A Linguistic Graph-Based Approach for Web News Sentence Searching

Kim Schouten and Flavius Frasincar

Erasmus University Rotterdam,
P.O. Box 1738, 3000 DR, Rotterdam,
The Netherlands
{schouten,frasincar}@ese.eur.nl

Abstract. With an ever increasing amount of news being published every day, being able to effectively search these vast amounts of information is of primary interest to many Web ventures. As word-based approaches have their limits in that they ignore a lot of the information in texts, we present Destiny, a linguistic approach where news item sentences are represented as a graph featuring disambiguated words as nodes and grammatical relations between words as edges. Searching is then reminiscent of finding an approximate sub-graph isomorphism between the query sentence graph and the graphs representing the news item sentences, exploiting word synonymy, word hypernymy, and sentence grammar. Using a custom corpus of user-rated queries and sentences, the search algorithm is evaluated based on the Mean Average Precision, Spearman's Rho, and the normalized Discounted Cumulative Gain. Compared to the TF-IDF baseline, the Destiny algorithm performs significantly better on these metrics.

1 Introduction

With news information volumes that are already overwhelming, a lot of the human mental activity is nowadays devoted to gathering, filtering, and consuming news information. Ingenious as we humans are, we have developed crude ways of filtering information quickly, using only a fraction of the time actually needed to process all the information accurately. However, reading only the titles or reading a text in a 'quick-and-dirty' fashion only goes so far. While the trade-off between speed and accuracy can probably not be broken due to the limitations of our brain, we do have the possibility to assist ourselves with tools in order to attain results that were previously impossible. One of these possibilities we would like to consider is to search for sentences, thus searching both within documents and across documents.

Despite its simplicity, experience has shown that methods based on TF-IDF [11], or similar metrics where a document is modeled as a bag of words, performs good in many circumstances. In our evaluation we therefore use TF-IDF as a baseline to test our approach against.

To better deal with the peculiarities of language, multiple approaches have been proposed where text is not regarded as a simple collection of words, but

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 57–64, 2013.

where, instead, text is processed using natural language processing techniques to extract valuable information that is implicitly encoded in its structure. A good example is the Hermes News Portal [12], where news items are annotated, linking found lexical representations to instances in an ontology. Queries, comprised of a selected set of concepts from this ontology knowledge base, can then be executed to get the news items pertaining to the entities in the query.

Unfortunately, there is an intrinsic problem with ontology-based approaches. Because not text, but the ontology is used for search, concepts that are not specified in the ontology cannot be found. An ontology thus makes the approach domain dependent. Furthermore, since an ontology is formally specified, information in the text has to be transformed to the logical form of the ontology. This is known to be difficult and sometimes even impossible since there are sentences that cannot be symbolized using first-order logic [1], and most ontologies use propositional logic or description logic (e.g., the many variants of OWL) which provides even less expressive power than first-order logic.

Thus, given that the meaning of language is both infeasible to extract and to represent, one can aim for a processing level just below the semantic interpretation phase [10]. Using the principles of homophonic meaning-representation and compositionality coming from the philosophy of language [3], we can represent language as an interconnected collection of disambiguated words, where the connections represent the grammatical relations that exist between words in a sentence. The interconnected collection of disambiguated words can then naturally be represented by a graph model, with nodes representing the words and edges representing the grammatical relations.

When both the news items and the sentence describing the user query are represented by the above type of graphs, the problem of searching for the query answers in the set of news items becomes related to the sub-graph isomorphism problem. While the standard graph isomorphism problem can be solved by Ullmann's algorithm [13], there are some additional considerations that prevent us from straightway applying it in this situation, the main one being that we are not only interested in full matches, but also in partial matches. We therefore want to measure the degree of similarity between the query sentence graph and all news item sentence graphs, and return a ranked list of sentences in the set of news items that are most similar to the user query.

2 The Destiny Framework

Based on the considerations in the previous section, our task is twofold: first, a process needs to be developed to transform raw text into a graph-based representation using the grammatical relations between words, and second, an algorithm needs to be devised that can compare graphs and compute a similarity score to enable ranking news sentences with respect to a query sentence. The Destiny framework is designed to incorporate both news item processing and query execution on the set of processed news items.

2.1 News Processing

To transform raw text into a dependencies-based graph representation, we have developed a pipeline consisting of various components with their own specific task. The same pipeline is used to process both news items and user queries. In Figure 1, a schematic overview of the framework is given. On the left hand side, the process of news item transformation is shown, while on the right hand side, the process of user query transformation and searching is depicted, each using the same processing pipeline in the middle.

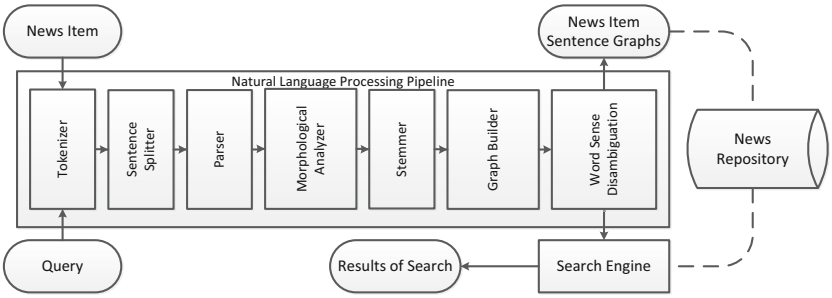


Fig. 1. Conceptual representation of framework

The tokenizer and sentence splitter, both standard components in the GATE framework [2], respectively, determine the word and sentence boundaries, after which the Stanford Parser [8] can extract the dependencies between words as well as the Part-of-Speech (POS) tags. Determining the lemma is then performed by the morphological analyzer, also a standard GATE component, and Porter's stemmer [9] algorithm is used to get the word stems.

Based on the information gathered so far, a graph representation of the sentence is built. To that end, we generate a node for each word, and connect these nodes with directed edges built from the syntactical dependencies output of the parser. Each node holds all information with respect to the word it represents: the literal word, lemma, stem, and POS tag. Each edge is labeled with the type of dependency that exists between these two nodes. As words in a sentence are uniquely represented by a node, the same word appearing multiple times in a sentence will result in multiple nodes, one node for each word.

Since words can have multiple senses, just comparing words based on their lexical representation will result in significant information loss. Therefore, the correct sense of each word has to be determined. This information is used in the search algorithm when determining synonym or hypernym relations between words. As the development of a word sense disambiguation algorithm is not the core topic of this paper, we have chosen to implement an existing one: the simplified Lesk algorithm [7].

2.2 News Searching

We devise a recursive algorithm for news searching, and the first important issue is to determine the starting point for the algorithm. As sentences can have different structures it is often not possible to compare two sentences by simply starting at the root. However, separately recursing from each node in the graph is not efficient. To combine efficiency with accuracy, only nouns and verbs in the query sentence graph are selected as they are the most probable good fits, the intuition being that these words are most rich in information compared to other types of words. Each one of these nodes in the query graph is used to find similar nodes in the news item graphs using an index on the stem values of all news item nodes. For each matching combination of a node in the query graph and a similar node in the news item graph, the recursion is performed with that combination as its starting point. All scores from the various recursion runs are aggregated on a sentence level so that to each combination of query and news item sentence, only the highest score is associated.

The algorithm compares the two graphs by starting to compare the two nodes in the query graph and a news item graph, respectively. After that, it will compare the nodes they are linked to, recursively traversing the graph until the comparison is complete. Because a node can have multiple parents and multiple children, the algorithm will try to compare edges for both directions. It is however constrained to only compare edges having the same direction.

There are multiple parameters in this algorithm that can be optimized, for example the various features that are used to assign similarity scores to nodes and edges. This optimization has been done using a genetic algorithm which will be explained later in this section. Besides feature weights, another parameter is used to control whether the recursion should continue in a given direction. If there is no direction that has a score (i.e., the similarity score of the edge and the node it connects to, but not any further in the graph) higher than the threshold as given by the parameter, than the algorithm will not recurse any further here.

Both the comparison of nodes and edges contributes to the total similarity score. With features weighted by the genetic optimization, the similarity score of a node or edge is the sum of all matching features. While edges only have one feature, the label denoting the type of grammatical relation, nodes have multiple features.

The five basic features are Boolean comparators on stem, lemma, the full word (i.e., including affixes and suffixes), basic POS category, and detailed POS category. The basic POS category consists of categories like noun, verb, adjective, etc., while the detailed POS category uses inflections like verb tenses and noun number (singular or plural) to create more fine-grained POS categories. These rather simplistic comparators are complemented by a check on synonymy and hypernymy using the senses acquired by the word sense disambiguation component and WordNet. When words are in the same broad POS category (e.g., nouns, verbs, adjective, and adverbs), but do not have the same lemma, they are possibly semantically related by synonymy or hypernymy.

For synonymy, WordNet is used to check whether both words with that particular sense are in the same synset, where a synset is used by WordNet to group words with the same meaning. For hypernymy, WordNet is used to find a relation of generalization between the two words. Such a relation does not have to be direct, but can also be indirect, involving multiple words as intermediary steps. For example, 'entity' is the most generalized form of 'car', even though there are multiple words between 'entity' and 'car' that are thus specializations of 'entity' but still generalizations of 'car'. As the strength of the generalization is dependent on the length of the path between the two words, we use the heuristic that the score as determined by the genetic algorithm is divided by the length of the shortest hypernymy path.

The last step in computing the similarity score of a node is an adjustment factor, which is a value between zero and one denoting how regular this word is in the news database. Being the inverse of the number of occurrences of this word in the database, the factor will be zero for the most common word, one for a word which occurs only once, and somewhere in between for the other cases.

2.3 Genetic Optimization

In order to optimize all parameters, a genetic algorithm was employed to attain good parameter settings. In Table 1, an overview is given of all parameters and of the weights assigned to it by the genetic algorithm when applied on the training set. All weights have values in the interval of 0 and 255. The fitness function the genetic algorithm maximizes is the normalized Discounted Cumulative Gain, one of the metrics used in the evaluation. The genetic algorithm itself is a standard implementation, using random initialization, cross-over, mutation, and elitist selection.

Interestingly, there seem to be many local optima for this specific optimization task, as the various cross-validation folds of the genetic algorithm, while roughly resulting in the same performance, yielded very different weights. The many local optima are probably due to the redundancy in natural language. For example, much of the same information can be found in both the literal word, as well as in the stem and the lemma. The same is true of a word and its context. Given the context, one is often able to guess a missing word, which means that much of the information in that word is already covered in the rest of the sentence. Given this redundancy, it is of no surprise that the features available to the genetic algorithm have a non-zero correlation.

In spite of the high standard deviations on the set of weights acquired by the genetic algorithm, there are still some interesting patterns that can be discovered. For example, we can see that the genetic algorithm clearly prefers the basic POS tags over the detailed ones. Also interesting is the fact that the fully inflected word is preferred over the lemma, which in turn is preferred over the stem of the word. While the fully inflected word represents more information than the lemma, just like the lemma represents more information than the stem, using the inflected form of a word makes it less generalizable than using the stem or even the lemma of a word.

Table 1. Optimized features and their weights

feature	one set of weights	**avg** weights	**stdev** weights
search threshold	12.44	95.64	67.42
node: word	182.11	125.30	65.75
node: lemma	50.24	105.11	67.53
node: stem	78.18	82.54	76.20
node: basic POS tag	206.46	119.00	77.96
node: detailed POS tag	2.84	50.62	59.76
node: synonym	106.20	136.90	71.53
node: hypernym	232.88	129.65	64.97
edge: label	200.69	171.82	71.58
significance factor importance	217.32	109.78	70.68

3 Evaluation

The results of Destiny are evaluated against a user-defined standard and compared with a classical text-based search algorithm. To that end, we have created a news database, containing 1019 sentences, originating from 19 news items, and 10 query sentences. Queries are constructed by rewriting sentences from the set of news item sentences. In rewriting, the meaning of the original sentence was kept the same as much as possible, but both words and word order were changed (for example by introducing synonyms and swapping the subject-object order). Each combination of a news sentence and a query was rated by at least three different persons. This led to a data set of over 30,500 data points (10 queries $\times$ 1019 sentences $\times$ at least 3 scores).

The various parameters of the search engine were optimized using a genetic algorithm, as discussed in the previous section. Therefore, the data set was split into a training and a test set. Because the results for each query are rather limited, the data set was split on the query level: 5 queries and their results went into the training set and the other 5 queries comprised the test set. To allow for cross-validation, 32 different splits were made. Because some queries have more results than others, all splits were checked to be balanced in the number of query results. In this way, the quantity of search results could not influence the quality of the results.

The baseline to compare our results against is the standard TF-IDF algorithm. All TF-IDF sores are computed over the test set for each of the 32 splits. As TF-IDF does not have to be trained, the training set was not used. Three metrics are used to compare the Destiny algorithm with the TF-IDF baseline, which are the Mean Average Precision (MAP), Spearman's Rho, and the normalized Discounted Cumulative Gain (nDCG) [6].

The main concern when using MAP is that it is Boolean with respect to the relevance of a certain sentence given a query, while the user scores return a gradation of relevance. This means that in order to compute the MAP, the user

scores have to split into two groups: relevant and not relevant, based on some cut-off relevance score c. Since this value is arbitrary, we have chosen to compute an average MAP score over all possible values of c, from 0 to 3 with a step size of 0.1.

The results, shown in Table 2, clearly show that Destiny significantly outperforms the TF-IDF baseline. For nDCG and Spearman's Rho, the p-value is computed for the paired one-sided t-test on the two sets of scores consisting of the 32 split scores for both Destiny and TF-IDF, respectively. For MAP, because we computed the average over all cut-off values, the same t-test is computed over 30 cut-off values $\times$ 32 folds which results in 960 split scores.

Table 2. Evaluation results

	TF-IDF mean score	Destiny mean score	rel. improvement	t-test p-value
nDCG	0.238	0.253	11.2%	< 0.001
MAP	0.376	0.424	12.8%	< 0.001
Sp. Rho	0.215	0.282	31.6%	< 0.001

4 Concluding Remarks

By developing and implementing Destiny, we have shown that it is feasible to search news sentences in a linguistic fashion. Using a natural language processing pipeline, both news items and queries are transformed into a graph representation, which are then compared to each other using the degree of sub-graph isomorphism as a measure for similarity. By representing text as a graph, the original semantic relatedness between words in the sentence is preserved, thus allowing the search engine to utilize this information.

Words, represented as nodes in the graph, are compared not only lexically, but also semantically by means of a word sense disambiguation algorithm present in the natural language processing pipeline. This allows for checks on both synonymy and hypernymy between words. The Mean Average Precision, Spearman's Rho, and normalized Discounted Cumulative Gain achieved by Destiny are significantly better than the scores obtained from the TF-IDF baseline.

As future work we would like to improve the accuracy of the search results by adding named entity recognition and co-reference resolution. The graph-based approach from [5] seems especially suitable for co-reference resolution. Furthermore, we would like to investigate the benefits of using a graph edit distance [4] measure, to mitigate problems with varying graph structures which are now not processed correctly.

Acknowledgement. The authors are partially supported by the Dutch national program COMMIT.

References

1. Barwise, J., Cooper, R.: Generalized Quantifiers and Natural Language. Linguistics and Philosophy 4, 159–219 (1981)
2. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M.A., Saggion, H., Petrak, J., Li, Y., Peters, W.: Text Processing with GATE (Version 6). University of Sheffield Department of Computer Science (2011)
3. Devitt, M., Hanley, R. (eds.): The Blackwell Guide to the Philosophy of Language. Blackwell Publishing (2006)
4. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
5. Haghighi, A., Klein, D.: Coreference Resolution in a Modular, Entity-Centered Model. In: Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2010), pp. 385–393. ACL (2010)
6. Järvelin, K., Kekäläinen, J.: Cumulated Gain-Based Evaluation of IR Techniques. ACM Transactions on Information Systems 20(4), 422–446 (2002)
7. Kilgarriff, A., Rosenzweig, J.: English SENSEVAL: Report and Results. In: 2nd International Conference on Language Resources and Evaluation (LREC 2000), pp. 1239–1244. ELRA (2000)
8. Klein, D., Manning, C.: Accurate Unlexicalized Parsing. In: 41st Meeting of the Association for Computational Linguistics (ACL 2003), pp. 423–430. ACL (2003)
9. Porter, M.F.: An Algorithm for Suffix Stripping. In: Readings in Information Retrieval, pp. 313–316. Morgan Kaufmann Publishers Inc. (1997)
10. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall (2002)
11. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill (1983)
12. Schouten, K., Ruijgrok, P., Borsje, J., Frasincar, F., Levering, L., Hogenboom, F.: A Semantic Web-based Approach for Personalizing News. In: ACM Symposium on Applied Computing (SAC 2010), pp. 854–861. ACM (2010)
13. Ullmann, J.R.: An Algorithm for Subgraph Isomorphism. J. ACM 23(1), 31–42 (1976)

Composite Patterns for Web API Search in Agile Web Application Development

Devis Bianchini, Valeria De Antonellis, and Michele Melchiori

Dept. of Information Engineering University of Brescia,
Via Branze, 38 - 25123 Brescia, Italy
{bianchin,deantone,melchior}@ing.unibs.it

Abstract. The ever growing availability of Web APIs enables web designers to aggregate APIs for fast development of new mashups. However, a web designer may also use Web APIs in different development tasks, like completion of an existing mashup, or substitution of one or more Web APIs within it. A system that supports the web designer during Web API selection should behave differently according to the development scenario the web designer is acting in. In this paper, we propose composite patterns which include different perspectives on Web API descriptions in order to serve distinct mashup development scenarios. Moreover, we provide a framework of tools, techniques and mechanisms to support the web designer during Web API selection according to the composite patterns.

1 Introduction

Agile development of web applications found new lifeblood in the context of web mashups [1,2]. Mashup development is fueled by on-line repositories, which offer large catalogues of Web APIs to be selected. For example, consider a web designer who aims at building a web application for storing and sharing her pictures. She may decide to use already available Web APIs, such as the `Amazon S3` or `MySpace` APIs, avoiding a time-consuming and costly implementation of their functionalities from scratch. Current repositories enable basic Web API search modalities (e.g., by keywords and categories), Web API filtering according to some features like the preferred protocol and data format, Web API ranking according to their popularity, that is, the number of mashups which include them. However, on current Web API repositories a web designer cannot perform more advanced searches: (i) by specifying both features of the Web API to search for and of the web mashup where the Web API will be used in; (ii) by exploiting semantic tagging, to avoid limitations of traditional tag-based search, specifically polisemy and homonymy; (iii) by relying on past choices and Web API ratings made by other designers (that is a specific kind of *collective knowledge* [3] applied in the mashup context); (iv) being assisted in more articulated development scenarios, other than the creation of a new mashup, such as the completion of an existing one, or the substitution of one or more Web APIs. For instance, if the designer has already chosen the `Amazon S3` API, the system should suggest her

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 65–72, 2013.

other Web APIs that are more commonly combined with `Amazon S3` for picture sharing and storage applications.

Other works rely on the collective knowledge coming from Web API use in existing web mashups [4,5], but the idea of adapting these techniques to different scenarios is still an unexplored field. As underlined in [6], this kind of approaches may suffer from the cold start problem, that is, if the criteria for Web API selection only relies on past mashups, a component that has been used in a significative number of cases gets more and more used despite the inner quality of the Web API. To avoid this shortcoming, in [6] different aspects to search for relevant Web APIs and for ranking the search results are balanced. Nevertheless, technical features are not considered, in particular for the purpose of tuning selection mechanisms according to the different search scenarios. The same limitations hold for approaches described in [7,8].

In this paper, we provide a model of composite patterns for Web API search. Specifically, a composite pattern is composed of a set of *metrics*, to be properly combined for matching the request against available Web APIs and ranking of search results. Metric aggregation may depend on the *search target*, such as the development of a new mashup, the completion of an existing mashup or the substitution of a Web API in a given mashup, considering only the categories, tags or features specified in the request or proactively suggesting search results according to the *collective knowledge*. The proposed metrics which compose the patterns have been implemented in a framework that is compliant with the `ProgrammableWeb` repository, a well known on-line catalogue where Web API providers share their own Web APIs and web designers can look for Web APIs they need. Our framework is partially based on the information extracted from the repository and on the collaborative semantic tagging system proposed in [7], where web designers can take actively part in the semantic tagging of Web APIs. In [9] we demonstrated the usefulness of considering multi-perspective Web API description for searching purposes on the `ProgrammableWeb` repository. In this paper, we perform a step forward with respect to [7,9], through the formulation of composite patterns.

The paper is organized as follows. Section 2 contains some preliminary definitions about the Web API model we rely on. Search patterns are described in Section 3. Section 4 closes the paper.

2 Preliminary Definitions

2.1 Web API Model

Figure 1 shows the UML representation of the Web API model we rely on. The model starts from Web API characterization presented in [7] and extends it with Web API technical features such as protocols and data formats, extracted from the `ProgrammableWeb` repository, that provides methods for retrieving basic information on Web APIs and mashups (`http://api.programmableweb.com/`). The repository can be kept updated by loading new Web API descriptions, while the designer may interact with our framework based on the model to exploit

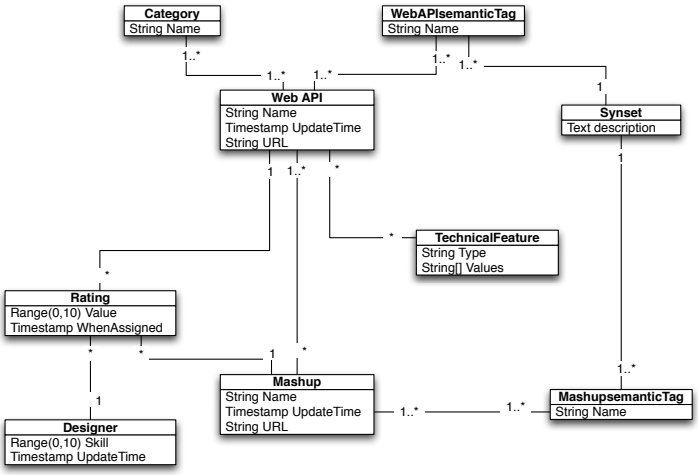


Fig. 1. The UML representation of the Web API model

additional facilities, such as semantic tagging, and apply advanced Web API search patterns, described in Section 3.

In our model, a Web API $\mathcal{W}$ is described by: (i) a name $n_{\mathcal{W}}$; (ii) a unique $URI_{\mathcal{W}}$; (iii) a category $c_{\mathcal{W}}$; (iv) a set $\{t_{\mathcal{W}}\}$ of semantic tags; (v) a set $T_{\mathcal{W}}$ of technical features, where a feature is a pair $\langle$`type`,{`values`}$\rangle$ (e.g., $\langle$`protocol`, {`SOAP`,`REST`}$\rangle$).

When a new semantic tag is assigned to the Web API, starting from the tag name specified by the designer, the WordNet lexical system is queried, all the synsets that contain that term are retrieved, thus enabling the designer to select the intended meaning. Therefore, a semantic tag $t_{\mathcal{W}}$ is a triplet, composed of: (i) the name of the term, extracted from WordNet; (ii) the synset, that is, the set of all the synonyms; (iii) the human readable definition associated with the synset (attribute `description` of the `synset` class in figure). The semantic tagging procedure has been extensively described in [7]. The set $T_{\mathcal{W}}$ of technical features is general enough to be adapted for different repositories. In particular, in the current version of the framework that is compliant with `ProgrammableWeb` repository, technical features are distinguished among the protocols adopted by the Web API (e.g., REST, SOAP, XML-RPC), the data formats (e.g., XML, JSON), the SSL support for security purposes and the authentication mechanism implemented within the Web API. For instance, the partial description of the `Amazon S3` API, classified in the `Storage` category, is the following:

$AmazonS3 = [URI_{AmazonS3}$: `http://aws.amazon.com/s3/`;

$t^1_{AmazonS3}$: ⟨`storage`, {`memory, computer memory, computer storage, store, memory board`}, "the process of storing information in a computer memory or on a magnetic tape or disk"⟩;

$\mathcal{P}_{AmazonS3}$: {`REST`, `SOAP`}; $\mathcal{F}_{AmazonS3}$: {`XML`}; $\text{SSL}_{AmazonS3}$: {`no`}; $\mathcal{A}_{AmazonS3}$: {`APIKey`}]

The model also considers a set $M_{\mathcal{W}}$ of mashups where the Web API has been included. Each mashup $m_{\mathcal{W}} \in M_{\mathcal{W}}$ is described by: (i) a name $n_{m_{\mathcal{W}}}$; (ii) a unique $URI_{m_{\mathcal{W}}}$; (iii) a set $\{t_{m_{\mathcal{W}}}\}$ of semantic tags which describe the mashup, defined in the same way as Web API semantic tags. For instance, the `Amazon S3` API has been included into 88 mashups by 57 developers[1] and, among them, the following ones:

$m_{Imaginalaxy} \in M_{AmazonS3}$ = [$URI_{Imaginalaxy}$: `http://www.imaginalaxy.com`;
$\{\mathcal{W}_{Imaginalaxy}\}$: {`Amazon S3, Facebook, Flickr, MySpace`};
$t^1_{Imaginalaxy}$: ⟨`photo`, {`photograph, exposure, picture, pic`}, "a representation of a person or scene in the form of a print or transparent slide or in digital format"⟩;
$t^2_{Imaginalaxy}$: ⟨`sharing`, {}, "using or enjoying something jointly with others"⟩]

$m_{tSpace} \in M_{AmazonS3}$ = [URI_{tSpace}: `http://totechphoto.com/space-light/`;
$\{\mathcal{W}_{tSpace}\}$: {`MediaFire, GoogleStorage, Dropbox, Amazon S3, MySpace`};
t^1_{tSpace}: ⟨`picture`, {`photograph, photo, exposure, pic`}, "a representation of a person or scene in the form of a print or transparent slide or in digital format"⟩;
t^2_{tSpace}: ⟨`sharing`, {}, "using or enjoying something jointly with others"⟩]

Finally, the model include a set $\mathcal{D}_{\mathcal{W}}$ of designers who used the Web API $\mathcal{W}$ to build their own mashups and a function $\mu_{\mathcal{W}} : \mathcal{D}_{\mathcal{W}} \times M_{\mathcal{W}} \mapsto [0..1]$ to represent a quantitative rating assigned to the Web API as used in one of the mashups, selected by the designer according to the NHLBI 9-point Scoring System. According to our model, each designer $d_i \in \mathcal{D}_{\mathcal{W}}$ owns at least one of the mashups in $M_{\mathcal{W}}$ and each mashup $m_{\mathcal{W}} \in M_{\mathcal{W}}$ has been developed by a designer in $\mathcal{D}_{\mathcal{W}}$. Each designer $d_i \in \mathcal{D}_{\mathcal{W}}$ is modeled through the skill $\sigma_i \in [0..1]$ for developing web applications. The skill is asked to the web designer during the registration to the system, according to a discrete scale: 1.0 for `expert`, 0.8 for `high confidence`, 0.5 for `medium confidence`, 0.3 for `low confidence` and 0.0 for `unexperienced`.

In the following, an expert designer d_1 assigned a very good rating to the `Amazon S3` API when used in the `Imaginalaxy` mashup, while a medium-level designer d_2 rated as excellent the same API when used in the `tSpace` mashup.

σ_{d_1} = 1.0 (`expert`); $\mu_{AmazonS3}(d_1, m_{Imaginalaxy})$ = 0.7 (`very good`)
σ_{d_2} = 0.5 (`medium`); $\mu_{AmazonS3}(d_2, m_{tSpace})$ = 0.8 (`excellent`)

2.2 Web API Request and Search Targets

A *Web API request* is defined as follows:

$$\mathcal{W}^r = \langle c^r_{\mathcal{W}}, \{t^r_{\mathcal{W}}\}, \{t^r_M\}, \{\mathcal{W}^r_M\}, T^r_{\mathcal{W}} \rangle \quad (1)$$

where $c^r_{\mathcal{W}}$ is the requested Web API category, $\{t^r_{\mathcal{W}}\}$ is a set of semantic tags specified for the Web API to search for, $\{t^r_M\}$ is a set of semantic tags featuring the mashup M where the Web API to search for should be used, $\{\mathcal{W}^r_M\}$ is the set of Web APIs already included in the mashup M and $T^r_{\mathcal{W}}$ is the set of required technical features.

[1] Last access on March 29th, 2013.

Not all the elements listed in Definition (1) are mandatory. Their specification within $\mathcal{W}^r$ depends on the *search target* pursued by the designer. We identified the following kinds of targets:

- *single Web API selection*, when the designer is developing a new mashup and aims at finding a Web API by specifying the Web API category $c^r_{\mathcal{W}}$ and semantic tags $\{t^r_{\mathcal{W}}\}$; optionally, the designer may also specify the desired technical features $T^r_{\mathcal{W}}$ for the Web API to search for;
- *advanced single Web API selection*, it is a variant of the previous search target; in this case, the designer has also in mind the kind of mashup where the Web API has to be aggregated, specified through a set of semantic tags $\{t^r_M\}$;
- *mashup completion*, when a mashup is already available and the designer desires to add a new Web API; in this case, $\{\mathcal{W}^r_M\}$ in the request is the set of Web APIs already inserted in the mashup under construction and, if the designer does not explicitly specify $T^r_{\mathcal{W}}$, it is automatically composed of the intersections of technical features of the Web APIs in $\{\mathcal{W}^r_M\}$, respectively; in fact, in this specific case, the best solution is that all the Web APIs within the mashup share the same protocol, data format and security features; the specification of $\{t^r_{\mathcal{W}}\}$ and $\{t^r_M\}$ are optional;
- *proactive mashup completion*, it is a variant of the previous search target; in this case, the designer does not specify the category $c^r_{\mathcal{W}}$ and the semantic tags $\{t^r_{\mathcal{W}}\}$ of the Web API to search for, since she does not know exactly what is available, what to look for and using what terms; therefore, she relies on suggestions of the system, which proposes candidate Web APIs based on collective knowledge coming from other existing mashups, which contain Web APIs close to $\{\mathcal{W}^r_M\}$ and present technical features similar to $T^r_{\mathcal{W}}$;
- *Web API substitution*, when the designer desires to substitute a Web API in an existing mashup; in this case, $c^r_{\mathcal{W}}$ and $\{t^r_{\mathcal{W}}\}$ are automatically extracted from the Web API to substitute; the construction of $T^r_{\mathcal{W}}$ follows the same rationale of the (proactive) mashup completion.

An example of request to find a Web API in the category `Storage` to be used for picture sharing together with the `Amazon S3` and `MySpace` APIs (*mashup completion* request) can be expressed through the category $c^r_{\mathcal{W}}$ = `Storage`, the semantic tags $\{t^r_M\}$ = {`picture`, `sharing`} (while $\{t^r_{\mathcal{W}}\} = \emptyset$), the set $\{\mathcal{W}^r_M\}$ = {`AmazonS3, MySpace`}.

3 Composite Patterns for Web API Search

A *pattern* for Web API search is defined through the combination of different similarity measures, specifically designed to compare categories, semantic tags, mashups and technical features in Web API descriptions and Web API request, and ranking criteria, to sort the searching results with respect to the specific

search target to be issued. The overall similarity is denoted with $Sim(\mathcal{W}^r, \mathcal{W})$ and is obtained as follows:

$$\begin{array}{l} Sim(\mathcal{W}^r, \mathcal{W}) = \omega_1 \cdot Sim_c(c^r_{\mathcal{W}}, c_{\mathcal{W}}) \ + \ \omega_2 \cdot Sim_t(\{t^r_{\mathcal{W}}\}, \{t_{\mathcal{W}}\}) + \\ \quad \omega_3 \cdot AGG_{m_{\mathcal{W}} \in M_{\mathcal{W}}}[Sim_t(\{t^r_M\}, \{t_{m_{\mathcal{W}}}\})] + \\ \quad \omega_4 \cdot AGG_{m_{\mathcal{W}} \in M_{\mathcal{W}}}[Sim_{comp}(\{\mathcal{W}^r_M\}, \{\mathcal{W}_{m_{\mathcal{W}}}\})] + \\ \quad \sum_{i=1}^{N} \omega_i \cdot Sim_i(\mathcal{W}^r, \mathcal{W}) \in [0..1] \end{array} \quad (2)$$

where the Web APIs returned as search results are those whose overall similarity is equal or greater than a threshold $\gamma \in [0..1]$, that can be set by the web designer (the highest the threshold, the highest the required overall similarity of search results with respect to the request). Specifically:

- $Sim_c(c^r_{\mathcal{W}}, c_{\mathcal{W}}) \in [0..1]$ is the similarity between the requested category and the category of $\mathcal{W}$;
- $Sim_t(\cdot) \in [0..1]$ is the similarity between two sets of semantic tags;
- $Sim_{comp}(\{\mathcal{W}^r_M\}, \{\mathcal{W}_{m_{\mathcal{W}}}\}) \in [0..1]$ is the mashup composition similarity between a mashup composed of a set $\{\mathcal{W}^r_M\}$ of Web APIs and another mashup composed of a set $\{\mathcal{W}_k\}$ of Web APIs;
- $Sim_i(\cdot) \in [0..1]$ is the similarity between $\mathcal{W}^r$ and $\mathcal{W}$ based on the i-th technical feature, such as protocols or data formats;
- $\omega_1 + \omega_2 + \omega_3 + \omega_4 + \sum_{i=1}^{N} \omega_i = 1$ are weights to be set according to the search target, as summarized in Table 1, where the category similarity is weighted less than the other factors, since the category is only a coarse-grained entry point to look for Web APIs in existing repositories.

Table 1. The setup of Web API matching weights depending on the search target

Search target	**Request formulation**
Single Web API selection	$\omega_1, \omega_2, \omega_i$ $(i = 1..N)$ equally weighted or $\omega_1 = 0.2$ and $\omega_2 = 0.8$ if $T^r_{\mathcal{W}} = \emptyset$
Advanced single Web API selection	$\omega_1 = 0.1, \omega_2, \omega_3, \omega_i$ $(i = 1..N)$ equally weighted or $\omega_1 = 0.2$ and $\omega_2 = \omega_3 = 0.4$ if $T^r_{\mathcal{W}} = \emptyset$
Mashup completion	$\omega_1 = 0.1, \omega_2, \omega_4, \omega_i$ $(i = 1..N)$ equally weighted
Proactive mashup completion	ω_4, ω_i $(i = 1..N)$ equally weighted
Web API substitution	$\omega_1 = 0.1, \omega_2, \omega_4, \omega_i$ $(i = 1..N)$ equally weighted

The similarity measures in Equation (2) can be implemented according to different techniques. Our patterns do not commit any specific solution and our framework implements several variants. For instance, category similarity may be based on the Jaro-Winkler string similarity measure between category names, or it may be even simpler, that is, $Sim_c(\mathcal{W}^r, \mathcal{W}) = 1$ if $c_{\mathcal{W}} = c^r_{\mathcal{W}}$, zero otherwise, or two categories may be considered as more similar as the number of Web APIs that are categorized in both of them increases with respect to the overall number of Web APIs classified in $c^r_{\mathcal{W}}$ and in $c_{\mathcal{W}}$ (in the latter case, the Dice coefficient is applied [10]). The similarity between two sets of semantic tags may be computed by applying the Jaro-Winkler string similarity measure between tag names or by considering the WordNet-based similarity described in [7]. The similarity between mashups may be computed as the number of common Web APIs in the

two mashups with respect to the overall number of Web APIs, through the Dice coefficient. Finally, the similarity between the technical features required in $\mathcal{W}^r$ and the ones of a Web API $\mathcal{W}$ may be computed as the number of common values for the same type of technical feature specified for $\mathcal{W}^r$ and $\mathcal{W}$.

The distinctive features of our framework rely on the aggregation function $AGG_{m_\mathcal{W} \in M_\mathcal{W}}[\cdot]$ and the ranking criteria. The aggregation function is applied in order to combine the different $Sim_t(\cdot)$ and $Sim_{comp}(\cdot)$ computations for all the mashups $m_\mathcal{W} \in M_\mathcal{W}$ in which the Web API $\mathcal{W}$ has been used. Such aggregation function can be chosen between `MAX` (the maximum $Sim_t(\cdot)$ and $Sim_{comp}(\cdot)$ values are chosen, respectively), `AVG` (the average values for $Sim_t(\cdot)$ and $Sim_{comp}(\cdot)$ are computed) or `DS` (weighting based on designers' skills). The last option takes into account that a Web API could be used in different mashups by designers $d_i \in \mathcal{D}_\mathcal{W}$ who have different skills σ_i. Therefore, the resulting formula is the following:

$$AGG_{m_\mathcal{W} \in M_\mathcal{W}}[Sim_t(\cdot)] = \frac{\sum_{i=1}^{|M_\mathcal{W}|} \sigma_i \cdot Sim_t(\{t_M^r\}, \{t_{m_\mathcal{W}}\})}{|M_\mathcal{W}|} \in [0..1] \tag{3}$$

where σ_i is the skill of the developer who owns the i-th mashup of $M_\mathcal{W}$. This formula ensures that the past experiences of more expert designers have a higher impact on the $Sim_t(\cdot)$ computation. Intuitively, the closest the σ_i and $Sim_t(\cdot)$ values to 1 (maximum value) for all the designers d_i, the closest the value of Equation (3) to 1.0. The computation of $AGG_{m_\mathcal{W} \in M_\mathcal{W}}[Sim_{comp}(\cdot)]$ is performed in the same way.

Finally, search results are ranked according to the following equation:

$$\rho(\mathcal{W}) = Sim(\mathcal{W}^r, \mathcal{W}) \cdot \prod_{j=1}^{n} [\rho_j(\mathcal{W})] \in [0..1] \tag{4}$$

where $\rho_j(\mathcal{W}) \in [0..1]$ are n ranking functions specified in the search pattern. In our framework, we consider common ranking functions related to Web API popularity (i.e., the number of mashups where the Web API has been used and the number of developers who adopted the Web API) and functions related to the novelty of Web APIs and their use (through the `timestamp` attributes in the model). Moreover, we also consider a novel ranking function based on the ratings assigned by web designers, consider as more important ratings the ones assigned by more expert designers:

$$\rho_1(\mathcal{W}) = \frac{1}{|\mathcal{D}_\mathcal{W}|} \cdot \sum_{i=1}^{|\mathcal{D}_\mathcal{W}|} \frac{\sum_{k=1}^{|M_k|} \sigma_i \cdot \mu_\mathcal{W}(d_i, M_k)}{|M_k|} \in [0..1] \tag{5}$$

where we must consider the ratings assigned by all the designers $d_i \in \mathcal{D}_\mathcal{W}$ who used the Web API $\mathcal{W}$ in mashups $\{M_k\}$.

4 Conclusions

In this paper we proposed a framework, compliant with the `ProgrammableWeb` repository, which implements a set of techniques and mechanisms to support

Web API search for agile web application development according to different patterns. Actually, the framework proposed in this paper suggests that there are really several aspects that can be mixed together in order to support Web API selection. Among them, also social relationships between developers could be used to automatically infer useful information, such as developers' reliability. Therefore, the Web API model could be further extended and the patterns modified as well. Additional information could be extracted also from other repositories [11], such as `Mashape` (`www.mashape.com`), which is a cloud API hub.

References

1. Ketter, W., Banjanin, M., Guikers, R., Kayser, A.: Introducing an Agile Method for Enterprise Mash-Up Component Development. In: Proc. of the 2009 IEEE Conference on Commerce and Enterprise Computing, pp. 293–300 (2009)
2. Rodríguez, R., Espinosa, R., Bianchini, D., Garrigós, I., Mazón, J.-N., Zubcoff, J.J.: Extracting Models from Web API Documentation. In: Grossniklaus, M., Wimmer, M. (eds.) ICWE 2012 Workshops. LNCS, vol. 7703, pp. 134–145. Springer, Heidelberg (2012)
3. Montanelli, S., Bianchini, D., Aiello, C., Baldoni, R., Bolchini, C., Bonomi, S., Castano, S., Catarci, T., De Antonellis, V., Ferrara, A., Melchiori, M., Quintarelli, E., Scannapieco, M., Schreiber, F., Tanca, L.: The ESTEEM platform: Enabling P2P semantic collaboration through emerging collective knowledge. Journal of Intelligent Information Systems 36(2), 167–195 (2011)
4. Greenshpan, O., Milo, T., Polyzotis, N.: Autocompletion for Mashups. In: Proc. of the 35th Int. Conference on Very Large DataBases (VLDB), Lyon, France, pp. 538–549 (2009)
5. Shafiq, M., Alhajj, A., Rokne, J.: On the social aspects of personalized ranking for web services. In: Proc. of 13th IEEE Int. Conference on High Performance Computing and Communications, pp. 86–93 (2011)
6. Torres, R., Tapia, B., Astudillo, H.: Improving Web API Discovery by leveraging social information. In: Proceedings of the IEEE International Conference on Web Services, pp. 744–745 (2011)
7. Bianchini, D., De Antonellis, V., Melchiori, M.: Semantic Collaborative Tagging for Web APIs Sharing and Reuse. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 76–90. Springer, Heidelberg (2012)
8. Dojchinovski, M., Kuchar, J., Vitvar, T., Zaremba, M.: Personalised graph-based selection of web aPIs. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 34–48. Springer, Heidelberg (2012)
9. Bianchini, D., De Antonellis, V., Melchiori, M.: A Multi-perspective Framework for Web API Search in Enterprise Mashup Design. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 353–368. Springer, Heidelberg (2013)
10. van Rijsbergen, C.J.: Information Retrieval. Butterworth (1979)
11. Bianchini, D., De Antonellis, V., Melchiori, M.: A Linked Data Perspective for Effective Exploration of Web APIs Repositories. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 506–509. Springer, Heidelberg (2013)

Evaluating the Interest of Revamping Past Search Results

Claudio Gutiérrez-Soto[1,2] and Gilles Hubert[1]

[1] Université de Toulouse, IRIT UMR 5505 CNRS,
118 route de Narbonne, F-31062 Toulouse cedex 9
{Claudio-Orlando.Gutierrez-Soto,Gilles.Hubert}@irit.fr
[2] Departamento de Sistemas de Información,
Universidad del Bío-Bío, Chile

Abstract. In this paper we present two contributions: a method to construct simulated document collections suitable for information retrieval evaluation as well as an approach of information retrieval using past queries and based on result combination. Exponential and Zipf distribution as well as Bradford's law are applied to construct simulated document collections suitable for information retrieval evaluation. Experiments comparing a traditional retrieval approach with our approach based on past queries using past queries show encouraging improvements using our approach.

1 Introduction

Information retrieval (IR) is finding information (usually text) that satisfies an information need from within large collections [12]. An information retrieval system (IRS) represents and stores large amounts of information in order to facilitate and accelerate determination of information estimated relevant to a user's query. A common IRS relies on two main processes: indexing and matching. Indexing intends to construct comparable representations of documents and queries. Matching intends to estimate the extend to which a given document is relevant to a query, usually represented via a score. Documents are then returned to the user in the form of list of results ranked by decreasing score.

IR systems emerged in the late 1940s but improvements really appeared from the late 1950s. Improvements first concerned indexing, ranking functions, weighting schemes, relevance feedback and then models [17]. IR improvements are highly related to evaluation of IRS dating back from the late 1950s [5]. The IR community notably benefited from TREC evaluation campaigns and workshops [21]. These have been offering researchers means to measure system effectiveness and compare approaches.

The literature of IR is crammed with different contributions such as indexing approaches, matching functions, formal models and relevance feedback approaches. However, few approaches gain advantage from searches performed in the past by previous users. Past searches constitute though a useful source of

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 73–80, 2013.

information for new users for example. For example, a user searching about a new subject could benefit from past searches led by previous users about the same subject.

The weak interest of IR in past queries may be understandable because of the lack of suitable IR collections. Indeed, most of the existing IR collections are composed of independent queries. These collections are not usable to evaluate approaches based on past queries since they do not gather similar queries for which ground thruth relevance judgments are provided. In addition, elaborating such collections is difficult due to the cost and time needed. An alternative is to simulate such collections. Simulation in IR dates back to at least the 1980s [10].

In this paper, on the one hand we propose an approach of creating simulated IR collections dedicated to evaluating IR approaches based on past searches. We simulated several scenarios under different probability distributions to build the collection of documents and determine the relevant documents given a query. On the other hand, we introduce a first approach of information retrieval using past queries based on result combination. We experimented this approach on different simulated IR collections using the aforementioned approach. We evaluated the effectiveness of our approach and compared it with a traditional retrieval approach.

This paper is organized as follows. In Sect. 2, related work on simulation in IR evaluation and the use of Past Searches in IR is presented. In Sect. 3 we present our approach to create a simulated IR collection described and the approach of retrieval based on past queries is presented in Sect. 4. Experiments are detailed in Sect. 5. Finally, conclusion and future work are presented in Sect. 6.

2 Related Work

2.1 Simulation in IR Evaluation

The use of simulation in IR is not recent, several approaches have been presented since 1980s. Simulation in IR can be seen as a method where a large collection of queries together with their judgments can be obtained without user interaction [10]. Simulation have been used in different contexts in IR. In [20], an algorithm was developed with the purpose to simulate relevance judgments of users. Here, simulated precision is compared with real precision. In [1], queries to find known-item are simulated. The authors proposed a model to build simulated topics that are comparable to real topics. Works dedicated to simulate the interaction among queries, click log, and preferences of users have been built [6,10]. In addition, today with the exponential growth of the Web, simulation provides interesting approximations of performance on the Web [2,13].

2.2 Past Searches in IR

Many approaches intending to improve results of queries using past searches can be found in the literature. A framework dedicated to improve effectiveness

measures such as Average Precision (AP), where a set of documents is assigned to the best system cluster (i.e. best answer given a query) can be found in [3]. Several approaches in IR use past queries for query expansion. For instance, similarity measures are defined in [16] to retrieve past optimal queries that are used to reformulate new queries or to propose the results of past optimal queries. [11] proposed to learn from old queries and their retrieved documents to expand a new submitted query. In [19], historical queries, in particular terms defining queries, are combined to improve average precision for new queries. In [18], feedback information, including clickthrough data and previous queries, are used to improve retrieval effectiveness. Another approach aiming to improve retrieval effectiveness is presented in [4]. A new selection technique using past queries is proposed to estimate utility of available information sources for a given user query. Here, relevance judgments of past queries were not used.

3 Simulating an Information Retrieval Collection

A usual IR collection is composed of three parts: a set of documents,a set of queries and a set of relevance judgments per query (i.e. indications on documents considered relevant or not relevant) [21]. Consequently, our approach aims at defining by simulation. Our method is split in two steps. The first step concerns the creation of terms, documents and queries. The second step involves the simulation of relevance judgments using Bradford's law.

3.1 Creation of Documents and Queries

In this first step, we use an alphabet in to build a set of terms. Each term is composed of letters of this alphabet. This set of terms can be split in subsets called topics in order to represent different subjects. Each letter is chosen using uniform distribution with the purpose to build a term. Thus, each term is unique.

In addition, each document is defined according to all the topics. In order to build a document, topics are selected using either the exponential or Zipf distribution and then terms constituting the document are chosen using uniform distribution. Thus, a document is constructed with terms from one topic mainly but not exclusively.

Past queries are built from documents. To built a past query, a document is chosen under uniform distribution. The terms that constitute the query are chosen from the document under uniform distribution. It is important to emphasize that the intersection among past queries is empty, that is, they have no terms in common. New queries are then built from past queries. For each past query a new query is built, either by changing or adding another term. Thus, the most similar query for the new query is its corresponding past query.

3.2 Simulating Relevant Judgments

In order to simulate the decision given by a user about if a document is relevant or not relevant for a given query, we relied on the zeta distribution. Zeta distribution

gives a discrete approximation of Bradford's law [8]. Bradford's law says that among the production of journal papers, there is an heterogeneous number of papers where most relevant papers are in few journals, while a few number of relevant papers are spread on a high quantity of journals. In our case, for a given query, it means that the most relevant documents should be at the top of the list (most relevant papers are in few journal), while a few relevant documents should be spread at down of the list document given a query.

In addition, we assume that for two very similar queries q and q', when a document is relevant for a query, it could be also relevant for the other query. In an intuitive way, there is a subset of common relevant documents for both queries. This does not implies that all relevant documents for query q, are relevant documents for query q'. With the objective to simulate this scenario, we use zeta distribution as follows. We retrieve documents for queries q' and q. Zeta distribution is applied to the set of common documents to the two queries to determine a subset of common relevant documents to q' and q. Eventually, zeta distribution is applied again to retrieved documents of each query q' and q (other relevant documents may be added), preserving the relevant common documents to q and q'. Therefore, the set of relevant documents for q' differs from the set of relevant documents for q.

4 Retrieval Using Past Queries

The basic idea behind of our approach is to incorporate to the system every query with its set of associated documents (query plus its list of documents, which are part of the answer of this query). Thus, the system has not only the set of documents but also the queries executed by user (past queries) with the set of associated documents them. At the beginning just there are documents without the queries, but every time a query is processed by the system, it is aggregated with its documents to the system. When a new query is submitted, first, it is checked and compared with the past queries in the system. When no similar query is found a traditional retrieval can be performed. When similar past queries are found, relevant documents of past queries can be used to build the result for the new query as well as combined with documents retrieved using traditional retrieval. Different strategies combining results from past queries and a traditional result. As first tested approach, we built the result for a new query by adding first the relevant documents from the most similar past query and then documents from a traditional retrieval.

5 Experiments

5.1 Experimental Environment

For this series of experiments, we used the English alphabet in order to build a set of terms. In a general way, the length of a term $|t|$ was between 3 and 7. This length was selected under uniform distribution. The number of terms $|T|$

was 700 for each experiment. We generated documents comprising between 300 to 900 words from a vocabulary composed of between 15 to 30 words for each document [9,15,14]. We defined 7 topics, each topic comprising 100 terms. When a document is built, terms of the other topics are chosen using either exponential distribution or Zipf distribution. Therefore, the most words to compose a document are chosen specific topic. For each experiment we generated five sets of documents comprising 700, 1400, 2100, 2800, and 3500 documents.

On the other hand, we defined 30 queries comprising between 3 to 8 terms for each experiment. The terms of a query were chosen from a particular document. Both terms and documents were chosen using uniform distribution to build the 15 past queries. It is important to mention that intersection among pairs of queries is empty.

In order to simulate judgments of users on documents retrieved given a query q, we implemented the zeta distribution with the purpose to represent the Bradford's law. We applied zeta distribution on the top 30 retrieved documents for each query. Each experiment gathers three different scenarios of zeta distribution for determining relevant documents by varying the parameter s with the values 2, 3, and 4.

5.2 Experimental Results

In this section, three experiments are detailed. We used exponential distribution to build the collection of documents D in the two first experiments and we used Zipf distribution [23] in the third experiment. We computed P@10 (precision at ten retrieved documents) on the results returned by our approach using past results and the traditional cosine retrieval, for each of the thirty queries. Then, we applied the Student's paired sample t-test to test if the difference between the two compared approaches with regards to P@10 was statistically significant.

Experiment 1. Exponential distribution with parameter equal to 1.5 was used to build the dataset D. When using zeta distribution with parameter $s = 2$ for relevance, our approach improved P@10 for 26 over 30 queries on average over the five sets of documents generated (i.e. comprising 700, 1400, 2100, 2800, and 3500 documents). The average P@10 improvement was +35.00 % over all queries. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00128 for the set of 700 documents. When using zeta distribution with parameter $s = 3$, our approach improved P@10 for 24.4 over 30 queries on average over the five sets of documents. The average P@10 improvement was +33.99 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00056 for 3500 documents. When using zeta distribution with parameter $s = 4$, our approach improved P@10 for 21.4 over 30 queries on average over the five sets of documents. The average P@10 improvement was +38.48 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00732 for 1400 documents.

Experiment 2. Exponential distribution with parameter equal to 1.0 was used to build the dataset D. When using zeta distribution with parameter $s = 2$ for relevance, our approach improved P@10 for 26.2 over 30 queries on average over the five sets of documents. The average P@10 improvement was +31.12 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00002 for 1400 documents. When using zeta distribution with $s = 3$, our approach improved P@10 for 21.6 over 30 queries on average. The average P@10 improvement was +26.10 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.01306 for 3500 documents. When using zeta distribution with parameter $s = 4$, our approach improved P@10 for 21.6 over 30 queries on average. The average P@10 improvement was +33.76 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00122 for 2800 documents.

Experiment 3. Zipf distribution with parameter equal to 1.6 was used to build the dataset D. When using zeta distribution with parameter $s = 2$ for relevance, our approach improved P@10 for 24.8 over 30 queries on average over the five sets of documents. The average P@10 improvement was +25.50 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00005 for 700 documents. When using zeta distribution with parameter $s = 3$, our approach improved P@10 for 22.6 over 30 queries on average. The average P@10 improvement was +22.77 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00034 for 1400 documents. When using zeta distribution with parameter $s = 4$, our approach improved P@10 for 24.8 over 30 queries on average. The average P@10 improvement was +27.92 %. Statistical significance was reached in all cases, the highest p-value for the Student's t-test being 0.00031 for 2100 documents.

5.3 Discussion

Due to space limitations, some experiment details were not presented, however additional observations are reported in this section. Using different parameters in zeta distribution ($s = 2, 3$ and 4) for relevance judgments allowed us to analyze how function influences on the average P@10. According to the evaluations we observed that average P@10 decreases for both compared approaches when the parameter s is increases in zeta distribution. In addition, we observed that there is not a radical tendency in the differences of average P@10 between the approach using past queries and the traditional retrieval approach when the number of documents increases.

Summarizing the results reported in this paper, each experiment and each scenario showed that the approach based on past queries always overcomes the traditional approach ($>$ +22.50 %). Statistical significance was always reached since the highest p-value for the Student's t-test was 0.01306.

In addition one should notice that we applied the Zipf distribution with value 1.6 only to simulate distribution frequencies of terms from topics for document creation. It would be interesting to test other Zipf distributions to simulate the

distribution of the frequencies of terms [23]. It is for this reason that we used not only the Zipf distribution but also exponential distribution to build collections of documents.

Eventually, our experiments used simulated datasets which allow us to have promising preliminary results. However, we should test other datasets using other distributions such as power law for term selection or using queries generated from different document collections.

6 Conclusion and Future Work

In this paper, we have presented on the one hand an approach of information retrieval using past queries. This approach is based on the reuse, for a new submitted query, of relevant documents retrieved for the most similar past query. On the other hand, due to the lack of available existing IR collections suitable for evaluating this kind of approach, we proposed an approach to creating simulated IR collections. We simulated several scenarios under different probability distributions to build the collection of documents and determine the relevant documents given a query. We experimented our approach of retrieval based on past queries on different simulated IR collections using the aforementioned approach. We evaluated the effectiveness of our approach and compared it with a traditional retrieval approach. Experiments showed encouraging results when evaluating the results for the top ten retrieved documents (P@10).

Future work will be devoted first to define more real evaluation datasets suitable for IR approaches based on past queries, adapting TREC collections for instance. We will also develop other approaches to construct the retrieved documents for a new query from various results of past queries, based on clustering methods [22] and based on diversification methods [7].

References

1. Azzopardi, L., de Rijke, M., Balog, K.: Building simulated queries for known-item topics: an analysis using six european languages. In: Proceedings of the 30th annual international ACM SIGIR, pp. 455–462. ACM, New York (2007)
2. Baeza-Yates, R., Castillo, C., Marin, M., Rodriguez, A.: Crawling a country: better strategies than breadth-first for web page ordering. Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, WWW 2005, pp. 864–872. ACM, New York (2005)
3. Bigot, A., Chrisment, C., Dkaki, T., Hubert, G., Mothe, J.: Fusing different information retrieval systems according to query-topics: a study based on correlation in information retrieval systems and trec topics. Inf. Retr. 14(6), 617–648 (2011)
4. Cetintas, S., Si, L., Yuan, H.: Using past queries for resource selection in distributed information retrieval. Tech. Rep. 1743, Department of Computer Science, Purdue University (2011), `http://docs.lib.purdue.edu/cstech/1743`
5. Cleverdon, C.W.: The evaluation of systems used in information retrieval (1958: Washington). In: Proceedings of the International Conference on Scientific Information - Two Volumes, pp. 687–698 (1959)

6. Dang, V., Croft, B.W.: Query reformulation using anchor text. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, WSDM 2010, pp. 41–50. ACM, New York (2010)
7. Drosou, M., Pitoura, E.: Search result diversification. SIGMOD Rec. 39(1), 41–47 (2010)
8. Garfield, E.: Bradford's Law and Related Statistical Patterns. Essays of an Information Scientist 4(19), 476–483 (1980), http://www.garfield.library.upenn.edu/essays/v4p476y1979-80.pdf
9. Heaps, H.S.: Information Retrieval: Computational and Theoretical Aspects. Academic Press, Inc., Orlando (1978)
10. Huurnink, B., Hofmann, K., de Rijke, M., Bron, M.: Validating query simulators: An experiment using commercial searches and purchases. In: Agosti, M., Ferro, N., Peters, C., de Rijke, M., Smeaton, A. (eds.) CLEF 2010. LNCS, vol. 6360, pp. 40–51. Springer, Heidelberg (2010)
11. Klink, S.: Improving document transformation techniques with collaborative learned term-based concepts. In: Dengel, A.R., Junker, M., Weisbecker, A. (eds.) Adaptive READ Research Project. LNCS, vol. 2956, pp. 281–305. Springer, Heidelberg (2004)
12. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (July 2008)
13. Marin, M., Gil-Costa, V., Bonacic, C., Baeza-Yates, R., Scherson, I.D.: Sync/async parallel search for the efficient design and construction of web search engines. Parallel Comput. 36(4), 153–168 (2010)
14. Silva de Moura, E., Navarro, G., Ziviani, N., Baeza-Yates, R.: Fast and flexible word searching on compressed text. ACM Trans. Inf. Syst. 18(2), 113–139 (2000)
15. Navarro, G., De Moura, E.S., Neubert, M., Ziviani, N., Baeza-Yates, R.: Adding compression to block addressing inverted indexes. Inf. Retr. 3(1), 49–77 (2000)
16. Raghavan, V.V., Sever, H.: On the reuse of past optimal queries. In: Proceedings of the 18th Annual International ACM SIGIR Conference, pp. 344–350. ACM, New York (1995)
17. Sanderson, M., Croft, W.: The history of information retrieval research. Proceedings of the IEEE 100(Special Centennial Issue), 1444–1451 (2012)
18. Shen, X., Tan, B., Zhai, C.: Context-sensitive information retrieval using implicit feedback. In: Proceedings of the 28th Annual International ACM SIGIR Conference, pp. 43–50. ACM, New York (2005)
19. Shen, X., Zhai, C.X.: Exploiting query history for document ranking in interactive information retrieval. In: Proceedings of the 26th Annual International ACM SIGIR Conference, pp. 377–378. ACM, New York (2003)
20. Tague, J.M., Nelson, M.J.: Simulation of user judgments in bibliographic retrieval systems. In: Proceedings of the 4th Annual International ACM SIGIR Conference, pp. 66–71. ACM, New York (1981)
21. Voorhees, E.M., Harman, D.K.: TREC: Experiment and Evaluation in Information Retrieval. MIT Press, Cambridge (2005)
22. Xu, R., Wunsch, D.I.: Survey of clustering algorithms. IEEE Transactions on Neural Networks 16(3), 645–678 (2005)
23. Zipf, G.K.: Human Behavior and the Principle of Least Effort. Addison-Wesley, Reading (1949)

Main Content Extraction from Web Documents Using Text Block Context

Myungwon Kim, Youngjin Kim, Wonmoon Song, and Ara Khil

Dept. of Computing, Soongsil University,
Seoul, Korea Republic
{mkim,liebulia,gtangel,ara}@ssu.ac.kr

Abstract. Due to various Web authoring tools, the new web standards, and improved web accessibility, a wide variety of Web contents are being produced very quickly. In such an environment, in order to provide appropriate Web services to users' needs it is important to quickly and accurately extract relevant information from Web documents and remove irrelevant contents such as advertisements. In this paper, we propose a method that extracts main content accurately from HTML Web documents. In the method, a decision tree is built and used to classify each block of text whether it is a part of the main content. For classification we use contextual features around text blocks including word density, link density, HTML tag distribution, and distances between text blocks. We experimented with our method using a published data set and a data set that we collected. The experiment results show that our method performs 19% better in F-measure compared to the existing best performing method.

Keywords: Web Document Analysis, Content Extraction, Tag Distribution, Block Distance, Context.

1 Introduction

As portable smart phones became widely distributed, users are able to access the Internet faster and more conveniently. Also, the development of new web standards and publishing tools enables the producers of the web documents to express their messages in easier and more diverse ways. The web documents with extremely free and diverse styles are generated in a fast speed not only by business organizations but also by individual users via web blogs or SNS. To provide users what they want promptly and precisely in the web service environment, where many types of web documents are increasing explosively, it is needed to precisely categorize, analyze, and understand web documents.

Recently, for this purpose, researchers have investigated automatic classification of the pre-designated areas of web documents such as the main content, advertisements, comments, menus and other. Moreover, studies have been actively conducted on auto-extraction of the main content of a web document which contains the most important information.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 81–93, 2013.

In the early studies, the researchers analyzed structural features of the HTML documents by reorganizing the HTML-based web documents in the form of DOM (Document Object Model) [1] tree structure and analyzed the results [2-4]. Recently, some alternative methods were introduced in order to extract the main content out of an HTML document based on its context, overcoming the limitation of the structural analysis of the HTML documents, such as: identifying the main content based on a linguistic model, which is obtained through learning emerging words and messages in the target document [5-8], and extracting the main content using distinct features such as tags and hyper-links in the target HTML document [9-12]. However, since all of these previous works tested their performances using the restricted types of data such as news and blogs, the applicability to the current web environment containing various types of documents, remains questionable.

In this paper, we propose a method that can complement the weaknesses of the existing methods. The proposed method extracts the main content of the target document more accurately, by decomposing the target HTML document into text blocks and determining each text block whether it belongs to the main content or not based on the contextual features of the text block such as HTML tag distribution around the text block and the information of its neighboring text blocks. The method is applicable to the actual web environment which is full of diverse types of documents.

2 Related Works

2.1 Classification/Extraction of the Main Content from Web Documents

The most representative method to extract the main content from a web document is the DOM tree structure analysis, using the structural features of the HTML documents [2-4]. Especially, in [2], they convert an HTML document into a DOM tree based on visually identifiable blocks for page separation and extraction of the main content block. In the DOM tree, the attributes of each node consist of its width, height, frequency of its appearance in the document. Then, the nodes are grouped based on their attributes and finally each group is determined whether it belongs to the main content or not based on the group features. The main purpose of this method is to investigate the possibility of extracting the main content automatically through machine learning without using any template information of each web page. However, since it only utilizes visibly identifiable information on the screen and structural features of the documents that are interpretable to a tree structure, when irrelevant information such as advertisement, spam message, and a list of articles constitutes the bigger part of the target document, its performance of extracting the main content degrades rapidly.

To overcome this weakness, other methods were proposed that utilize HTML tags and text messages as distinctive attributes on the pattern of the HTML document composition [5-8, 10-12]. [5] and [7] convert an HTML document into a list of tags and text tokens, analyze the order and context of those items using probability models and extract the document regions which match to one of the known patterns as the

main content of the document. [11] and [12] propose a process that groups the lines of the target document using the appearance frequencies of the HTML tags and non-tag texts, then decides the eligibility of each group as the main content. The bottom line of these methods is that they attempt to analyze the document on the basis of the text tokens and patterns, beyond the structural analysis. However, the use of the text tokens of documents causes a risk to constantly revise and expand the database for learning to handle newly published web documents.

Other methods that were recently introduced utilize the characteristics of the HTML documents as web documents, which differentiate them from others in extracting the main content [9, 13]. [9] focuses on the fact that the main content of web documents usually consists of texts. It first segments text blocks out of the target HTML document. Then it decides whether a text block belongs to the main content based on the attributes such as the number of words and hyper-linked words in the text block, and the information of its neighboring text blocks. By doing so, the method shows a relatively high performance. However, it was verified by only using limited source of data such as news or blogs. It is still questionable whether it is applicable in the current web environment where many different kinds of web documents exist.

2.2 Main Content Extraction Based on Word/Link Density in Text Blocks

The best known open system for main content extraction from web documents is Boilerpipe proposed by the L3C research center [9, 13]. Boilerpipe consists of four steps. The first step is to divide the target HTML document into text blocks using some empirical rules. In the second step, it derives attributes of each text block by calculating each text block's word density and link density corresponding to the number of words and links contained in the text block, respectively. The next step is to build a classification model by machine learning based on the attributes of each text block and the word densities and link densities of its neighboring text blocks. Finally, using the model it decides which text block is the main content of the target document.

To divide an HTML document into text blocks, Boilerpipe utilizes HTML tags. All HTML tags in the document become group separators and all text segments separated by them are configured into text blocks. The only exception is tag <a> because it is only used to insert the hyper-link information and does not bring any structural changes. Therefore, <a> tags are only used for calculating link density in an HTML document.

After separating all text blocks, for each text block TB_i, the word density ($D_{WORD}(TB_i)$) and link density ($D_{LINK}(TB_i)$) are calculated according to equations (1). In the equations, $Word(TB_i)$, $Sentence(TB_i)$ and $LinkedWord(TB_i)$ denote the set of all words, the set of all sentences and the set of hyper-linked words by <a> tag in the text block, respectively. [9] identifies sentences in a text block by the number of words, and each sentence is assumed to have 80 words.

$$D_{WORD}(TB_i) = \frac{|Word(TB_i)|}{|Sentence(TB_i)|}, \quad D_{LINK}(TB_i) = \frac{|LinkedWord(TB_i)|}{|Word(TB_i)|} \tag{1}$$

Each text block has six attributes for machine learning and classification including the word density and link density of itself and the word densities and link densities of its neighboring text blocks.

In [9], the authors identified all text blocks through the procedure described above in the L3S-GN1 data set, a collection of 621 news articles on the web gathered by Google search. They then checked each text block and marked whether it belonged to the main content of the article or not to build the data for machine learning. They measured the classification accuracy of the method with the decision tree based classification algorithm and the 10-fold cross validation. To verify the superiority of the method to others, they compared the result with other existing methods, and successfully achieved their goals. It implicates that the six attributes, which are attainable in a relatively short period of time can classify the main content more accurately. However, their verification procedure remains questionable because they took the accuracy of the non-main content classification into consideration when they calculated the accuracy of the main content classification. Since the non-main content take the larger part of the target data set, it is difficult to recognize that the reported accuracy represents the exact accuracy of the main content classification.

3 Main Content Extraction Using Text Block Context

In this section, we introduce a new method to extract the main content from HTML documents more accurately using the contextual features of text block. The contextual features consist of the HTML tag distribution around the text blocks and the neighboring text block information in addition to previously introduced and proven attributes, i.e. the word density and link density of the text blocks. Initially, following the procedure of [9], the target HTML document is divided into text blocks and the word density and link density are used as the attributes of each text block. The new method adds two additional aspects to the existing text block attributes for main content extraction: the HTML tag distribution and the neighboring text block information.

3.1 HTML Tag Distributions Around the Text Blocks

An HTML document is organized by predefined and tree-structured HTML tags. In the main content of an HTML document, which mostly consists of texts, HTML tags for the text format and paragraph separation are mainly used [14]. Based on these characteristics, we utilize parent tags, which are immediate upper-level HTML tags that contain text blocks in a tree structure, as one of the attributes to determine whether a given text block belongs to the main content or not.

Figure 1 is an example of parent tag extraction for some text blocks. Since a parent tag is an immediate upper-level tag that encompasses the given text block as mentioned before, the parent tag of "By", the first text line in the upper part of Figure 1, is <h3> and is the parent tag of "Associated Press". In the case of the text blocks in the lower part of Figure 1, the parent tag of "NASA astronaut Sunita..." and "Russian cosmonaut Yuri..." are the <p> tag which is located right above them. The

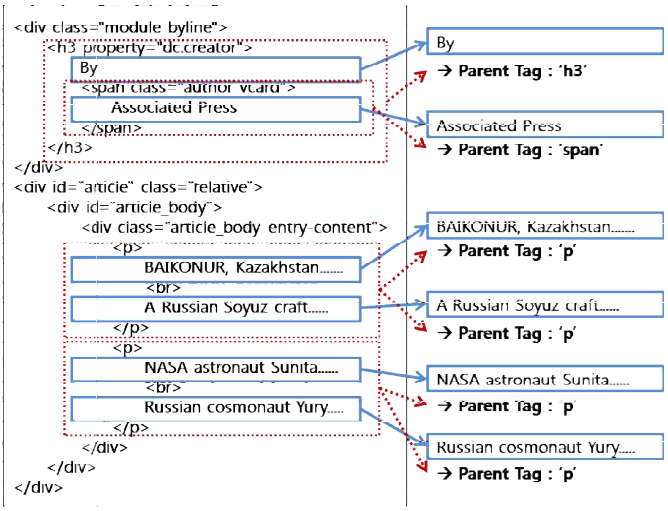


Fig. 1. Tag distribution around the text block: parent tag information

tag, located between these two text blocks cannot be the parent tag because it is a child tag of the <p> tag and it does not include these two text blocks.

Any HTML tag – total of 94 tags under the HTML standard 4.0.1 [14] - can be a parent tag of a text block. Among them, tags for text format processing and paragraphing are mostly used near the main content area of an HTML document. Therefore, in this paper, these tags which are frequently used around the main content are considered as the only candidate group of the parent tags. To enhance the effectiveness of the main content extraction process we propose, the syntactic and semantic information of all HTML tags [14] is checked, and their distribution and density information are analyzed with various examples in the Korean web such as news articles and blogs. Figure 2 shows the partial result of these examples.

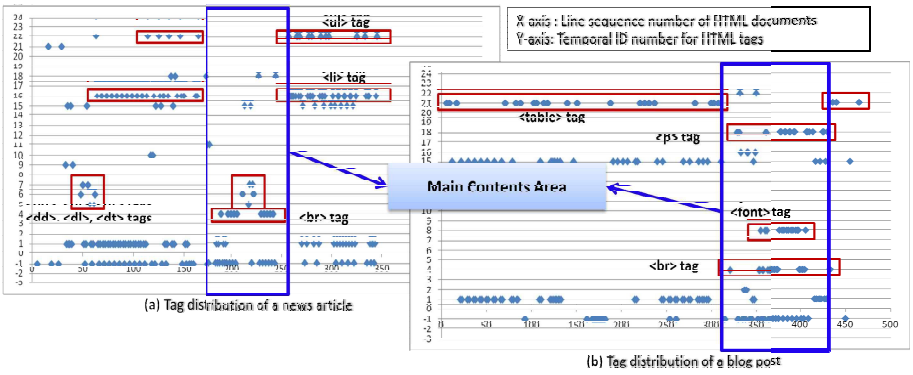


Fig. 2. Tag distribution of an HTML document

In the figure, the horizontal axis in graph (a) and (b) represents the sequence number of lines in the target document that is refined beforehand and the vertical axis represents the original identification number that was assigned to each HTML tag. For example, a point (x,y) on the two-dimensional graph – blue dots in the figure - represents that the HTML tag corresponding to the ID number 'y' appears in the 'x'-th line of the target document. With these conditions in mind, when you look at the graph (a), you can find that
, <dd>, <dl>, and <dt> mainly appear on the inside of the main content, while <ul>, <li>, and others are concentrated on the outside of the main content. (Similarly, in the graph (b),
, <font>, and <p> are located mainly on the inside of the main content.)

We conducted the same analysis as Figure 2 with 15 other arbitrarily collected HTML documents and defined 22 tags that mainly appeared on the inside of the main content area and less likely to appear in other areas. Table 1 is the list of 22 tags, and these tags were used for extracting parent tag information. When extracting the parent tags out of a document, if any extracted tag is one of those in Table 1, then it is separately marked. If not, it is classified into the same group as the rest. This procedure is necessary for learning a classification model during the machine learning process, focusing more on the main content text blocks than the non-main content text blocks and reducing the possibility of false learning by noises and irregularities.

Table 1. Meaningful HTML tags of the text block's parent tag information

Tag Name	Html Function/Description	Tag Name	Html Function/Description
a	Insert a hyper link	img	Insert an image
b	Make the next text bold	li	Describe lists in ,
blockquote	Indicate a quoted paragraph	ol	Produce an ordered list
br	Begin a new line	p	Paragraphing
dd	Describe definitions declared in <dl>	pre	Maintain document style
dl	List of definitions	q	Indicate a quoted word and phrase
dt	Describe terms declared in <dl>	table	Insert a table
font	Set font type	ul	Produce an unordered list
h1~h6	Heading (biggest size ~ smallest size)		

3.2 Neighboring Text Block Information

The second set of information for the main content extraction method we propose consists of two pieces of information: the neighboring text block information and the distance to the neighboring text blocks.

Since the main content of a web document shows up in one specific area of the whole document, texts in the main content are located closely in that area. It means if a text block is a part of the main content, its neighboring text block is likely to be as well. Also, if a text block is not a part of the main content, its neighboring text block is less likely to be so. Therefore, the distances from a given text block to its neighboring

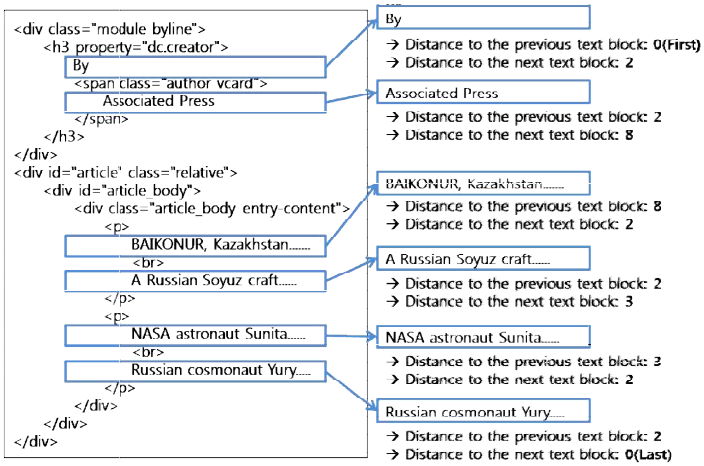


Fig. 3. Distance to the neighboring blocks

text blocks and the neighboring text block's attributes are essential in deciding whether a certain text block belongs to the main content or not. In this paper, we propose a method that can extract the main content more accurately based on such information.

The distance from the current text block to its neighboring text block is measured by the number of lines between the text blocks in the preprocessed HTML file with each tag separated on a line. An example is shown in Figure 3. The distance from the first text block to its preceding text block is set to 0. Similarly, the distance from the last text block to its following text block is also set to 0. In a sense the distance between text blocks represents the density of text blocks.

Additionally, the information of two preceding and two following text blocks is also utilized. As mentioned before, the main content text blocks tend to be located nearby from one another and so do the non-main content text blocks. To take advantage of such an observation in classifying each text block we use the information of its surrounding text blocks as the context information.

3.3 Attributes for Classification and Classification Algorithm

In this paper, as mentioned above, we propose a method based on machine learning for automatic extraction of the main content from HTML documents using the combination of previously introduced attributes and newly introduced ones in this paper.

Total of 25 attributes are used, and five attributes from each text block were extracted. The attributes introduced in [9] such as word density and link density, and the ones proposed in this paper such as parent tags, the distance to the preceding text block, and the distance to the following text block. Additionally, as mentioned in Section 3.2, for each text block the information of the two preceding text blocks and two following text blocks are used as the context information as well. Therefore, the set of attributes for a text block is composed of 25 attributes from five text blocks including its two preceding and two following text blocks.

Machine learning for classification is based on the decision tree model which is widely used. A decision tree is composed of nodes and arcs which represent attributes and attribute values, respectively. A decision tree is constructed by recursively dividing the data space along the selected attribute into subspaces in each of which classes are well separated. When selecting an attribute to divide the target data into subgroups of good class separation, different algorithms use one of these: Chi-square statistics, Gini index, or entropy index [15, 16]. In this paper, we used J48 method provided by WEKA [17]. It is most widely used methods and one of the decision tree based on C4.5 algorithm which uses the entropy index for attribute selection [16, 17].

4 Experiments and Evaluation

In Section 4, we describe attribute extraction through preprocessing of web documents, experimental data for learning and evaluation, learning algorithm, and performance evaluation. We also verify the proposed method by comparing it with the existing methods using both public and privately collected data.

4.1 HTML Document Preprocessing for Attribute Extraction

Before identifying each text block and extracting its attributes, it is necessary to refine the improperly written HTML documents. Although most HTML documents abide to the grammatical rules of HTML, some of them ignore the HTML codes that do not affect their final display on the screen. Since most HTML documents do not separate lines clearly, they have poor readability, thus, it is difficult to discern the text blocks and to calculate the distance between text blocks. Consequently, these documents should undergo a refining process that corrects or supplements their grammatically improper tags and separates lines to make each line to have only one HTML tag or one text block. The Neko parser [18], an HTML parser, was used for the refining process, and Figure 4 is a partial excerpt of a refined HTML document.

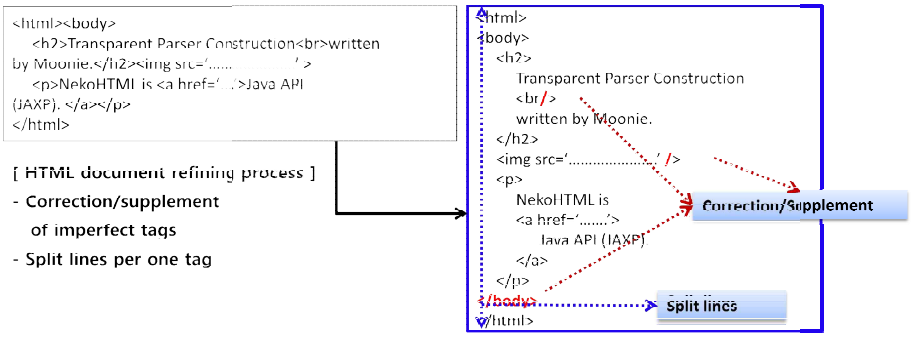


Fig. 4. HTML document refining process

After the refining process, following the method of [9], all text blocks are identified using all tags as separators except <a>, then the word density and link density for each text block are calculated. Finally, the attributes proposed by this paper, i.e. the surrounding tag distribution and the information of the neighboring text blocks, are extracted and added to the attribute vector of each text block along with the word and link densities.

4.2 Experiment Data

To verify the proposed method, we used two data sets in this paper.

The first data set is L3S-GN1 [19], a public data set made by a German research center (L3S) to compare and evaluate the performance of the main content extraction from web documents. Based on the assumption that the news articles published by various agencies are a generic document with diverse web contents, the authors gathered 621 English news articles through Google web search and published the data. In the data each text block is marked manually whether it belongs to the main content or not.

The second data set is called 965-GWeb [20], a collection of 965 web documents we gathered. It is a collection of web documents gathered through Google search like L3S-GN1, but for a wide spectrum of document types, we collected various kinds of documents such as news articles, blog posts, SNS pages, wiki documents, image/video pages, etc. Then, we checked and marked each text block manually whether it is a part of the main content or not.

The number and proportion of all text blocks and main content text blocks are shown in Table 2

Table 2. Distribution of the training data's records (text block) and class (main content and non-main content)

Data Set	Number of HTML Documents	Number of Learned Records (Text Blocks)		
		Total Records	Main Content (%)	Non-Main Content (%)
L3S-GN1 [19]	**621**	84,198 (Average 136/doc)	13,209 (15.69%) (Average 21/doc)	70,989 (84.31%) (Average 114/doc)
965-GWeb [20]	**965**	141,808 (Average 147/doc)	22,308 (15.73%) (Average 23/doc)	119,500 (84.27%) (Average 124/doc)

To use the both data sets shown in Table 2 for learning and evaluation without any separate validation data, all data in the data set are split for 10-fold cross-validation and then used for learning and evaluation.

4.3 Learning Algorithm and Performance Measure

For automatic extraction of the main content area, a classification method was used to determine whether a certain text block belongs to the main content or not. As for the

learning algorithm, J48 was used, which is a decision tree algorithm based on C4.5 [15, 16], included in WEKA [16, 17].

In the algorithm, two parameters are supported. The one is the reliability of the training data and the other is the number of minimum objects at the leaf node of tree. In this paper, while maintaining the reliability of the training data we collected and tested directly, as an effort to make the method remain applicable to the current web environment where new contents of various types are constantly increasing, we configured the confidence factor - 'confidenceFactor' option in J48 - 70% instead of 100%. As shown in Table 2, considering that the main content of one document consists of 20 text blocks on average, we set to 10 the number of minimum objects at the leaf node of decision tree - 'minNumObj' option in J48, which is about 50% of the average number of main content text blocks.

To measure the accuracy of the main content extraction, we used F-measure, which is the most widely used in the evaluation of classification systems. Normally, in multi-class classification evaluation, for the final F-measure to be calculated, the F-measure for each class is calculated first and then the F-measure for each class weighted by the size or significance of the class is summed over the whole classes. Since in this paper we have the two-class classification problem, i.e. the main content and the non-main content, we can acquire the final F-measure by summing up the F-measures for the main content and the non-main content, weighted by the number of data per class. But, even in a multi-class classification when the data are disproportionately distributed as shown in Table 2, the F-measure of the total data can be swayed by the F-measure of the larger class. For example, in Table 2, the F-measure of the main content is 0.5 and the F-measure of the non-main content is 0.9. Thus, the F-measure of both classes based on the number of data is 0.84, which is much higher than 0.5. This means that even with the accuracy of main content classification, the figure that matters is quite low. The final value is relatively high due to the influence of the larger and irrelevant class. In this paper, since the main purpose is to identify the main content area with accuracy, we only adopt the F-measure of the main content class to compare our method with the existing methods and eventually prove the validity of our method.

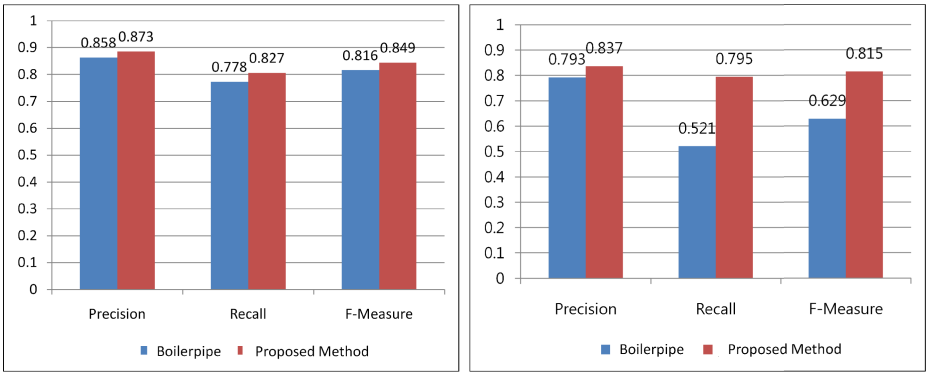
4.4 Performance Evaluation and Comparison

As mentioned in Section 2, according to [13], Boilerpipe [9] is the most efficient method for main content extraction. In this paper, we compared the proposed method with Boilerpipe using two data sets described in Section 4.2, However, since the reported performance of Boilerpipe is the result of multi-class classification with both classes of the main content and non-main content, to avoid the distortion resulting from any data disproportion, as mentioned in Section 4.3, the classification accuracy of Boilerpipe was re-calculated so that it only covers the main content class.

Table 3 and Figure 5 compare Boilerpipe and the proposed method in precision, recall, and F-measure, based on the two data sets.

Table 3. Performance comparison between Boilerpipe and the proposed method

	L3S-GN1 Data Set [19]			965-GWeb Data Set [20]		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Boilerpipe [9]	0.858	0.778	0.816	0.793	0.521	0.629
Proposed Method	0.873	0.827	0.849	0.837	0.795	0.815



(a) Performance comparison based on L3S-GN1 dataset (b) Performance comparison based on 965-GWeb dataset

Fig. 5. Performance comparison between Boilerpipe and the proposed method

As shown in the table and figure, the proposed method marked higher figures for all performance measures than Boilerpipe, known as the best performing method by far. For L3S-GN1, the F-measure of the proposed method is approximately 3% higher than Boilerpipe and 19% higher for 965-GWeb. The result proves the validity of the proposed method. Also, it shows that the proposed method can be more adaptable to various types of documents, because unlike Boilerpipe, the proposed method showed similar performance for both data sets.

Especially, the performance difference between Boilerpipe and the proposed method is significantly larger for the 965-GWeb data set than for the L3S-GN1 data set. The reason for this gap seems to be the fundamental difference between the two data sets. That is, L3S-GN1 is a collection of English news articles on the web, while 965-GWeb is a collection of web documents with various types and languages. It also means that the attributes such as word density and link density are effective for main content extraction of English news articles, while the proposed attributes of the neighboring text blocks have little contribution. However, the word density and link density may not be effective any more for the documents containing more diverse types of contents and languages, and the proposed attributes contribute significantly for main content extraction in this case.

5 Conclusion and Future Works

Extracting the main content, the important area of any document, out of the large group of web documents with free and diverse styles is a substantial and fundamental

task to understand and analyze web documents better to achieve improved services. For this purpose, we proposed a new method to automatically extract the main content of a web document using the contextual features of text blocks. Through an experiment, we proved that the proposed method showed a better performance in F-measure than the currently known best performing method by 3% when applied to a publicly available data set and 19% when applied to the privately collected data set. The proposed method showed relatively similar performance for both data sets. The experiment result shows that the proposed method is efficient and more suitable for main content extraction in the current web environment.

We need future work to supplement the proposed method.

The method needs to be extended to automatically extract from web documents various document areas which users are interested in, including comments, advertisements, article lists or menu. For this, we need to find out more general attributes that are applicable to different areas of web documents. In addition, a study for performance enhancement of the main content extraction should be conducted. In this paper, we employed decision tree for classification. However, for enhanced performance an ensemble classifier should be investigated to integrate different classifiers.

Acknowledgments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant no. 2011-0010719)

References

1. (Februry 2013), http://en.wikipedia.org/wiki/Document_Object_Model
2. Deng, C., Shipeng, Y., Ji-Rong, W., Wei-Ying, M.: VIPS: a Vision-based Page Segmentation Algorithm. Microsoft Technical Report(MSR-TR-2003-79) (2003)
3. Suhit, G., Gail, E.K., David, N., Peter, G.: DOM-based Content Extraction of HTML Documents. In: 12th International Conference on World Wide Web, pp. 207–214 (2003)
4. Suhit, G., Gail, E.K., Peter, G., Michael, F.C., Justin, S.: Automating Content Extraction of HTML Documents. World Wide Web 8(2), 179–224 (2005)
5. Jeff, P., Dan, R.: Extracting Article Text from the Web with Maximum Subsequence Segmentation. In: The 18th International Conference on World Wide Web, pp. 971–980 (2009)
6. Stefan, E.: A lightweight and efficient tool for cleaning Web pages. In: The 6th International Conference on Language Resources and Evaluation (2008)
7. Stefan, E.: StupidOS: A high-precision approach to boilerplate removal. In: Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop, pp. 123–133 (2007)
8. Young, S., Hasan, J., Farshad, F.: Autonomic Wrapper Induction using Minimal Type System from Web Data. In: Artificial intelligence, pp. 130–135 (2005)
9. Christian, K., Peter, F., Wolfgang, N.: Boilerplate Detection using Shallow Text Features. In: The Third ACM International Conference on Web Search and Data Mining, pp. 441–450 (2010)

10. Jian, F., Ping, L., Suk Hwan, L., Sam, L., Parag, J., Jerry, L.: Article Clipper- A System for Web Article Extraction. In: 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 743–746 (2011)
11. Tim, W., William, H.H., Jiawei, H.: CETR - Content Extraction via Tag Ratios. In: 19th International Conference on World Wide Web, pp. 971–980 (2010)
12. Tim, W., William, H.H.: Text Extraction from the Web via Text-to-Tag Ratio. In: The 19th International Conference on Database and Expert Systems Application, pp. 23–28 (2008)
13. (July 2012), http://tomazkovacic.com/
14. W3C (February 2013), http://www.w3.org/TR/html401/
15. Jiawei, H., Micheline, K.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2006)
16. Ian, H.W., Eibe, F.: Data Mining: Practical Machine Learning Tools and Techniques. Elsevier (2005)
17. Waikato Univ. (February 2013), http://www.cs.waikato.ac.nz/ml/weka/
18. Andy, C., Marc G.: (February 2012), http://nekohtml.sourceforge.net/
19. L3S Research Center (February 2013), http://www.l3s.de/~kohlschuetter/boilerplate/
20. (February 2013), http://121.78.244.168:8090/ice/index.jsp

A Similarity-Based Approach for Financial Time Series Analysis and Forecasting*

Marcos Vinicius Naves Bedo[1], Davi Pereira dos Santos[1], Daniel S. Kaster[2], and Caetano Traina Jr.[1]

[1] Inst. of Math. and Computer Science - University of São Paulo (USP) - Brazil
{bedo,davips,caetano}@icmc.usp.br
[2] Dept. of Computer Science - University of Londrina (UEL) - Brazil
dskaster@uel.br

Abstract. Financial time series analysis have been attracting research interest for several years. Many works have been proposed to perform economic series forecasting, however, it still is a hard endeavor to develop a general model that is able to handle the chaotic nature of the markets. Artificial intelligence methods such as artificial neural networks and support vector machines arose as promising alternatives, but they hide the processing semantics, limiting the result interpretation. In addition, one of the main drawbacks of the existing solutions is that they usually cannot be easily employed as building blocks of new analysis tools. This paper presents a new approach to financial time series forecasting based on similarity between series patterns using a database-driven architecture. We propose a new feature extractor based on visual features associated with a boosted instance-based learning classifier to predict a share's behavior, thus improving the human analyst understanding and validation of the results. The analysis is defined through extended SQL instructions and executed over a fast and scalable engine, which makes our solution adequate to provide data analysis support for new applications handling large time series datasets. We also present experiments performed on data obtained from distinct market shares. The achieved results show that our approach outperformed existing methods in terms of accuracy, running time and scalability.

Keywords: Financial time series, Time series forecasting, Similarity retrieval, Classifiers Methods.

1 Introduction

The stock market, based on real time transactions, is a new and important component of the contemporary capitalist financial system. It allows a company to obtain investments, take strategic decisions according to the opinion of the

* This research was supported by the scholarship grants from the São Paulo State Research Foundation (FAPESP) and by the Brazilian National Research Council (CNPq) and by CAPES.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 94–108, 2013.

partner-owners, and expand its activities on a global level through investors around the world. In recent years, the number of shares traded on the stock exchange markets have grown tremendously [12], and expectations point to an even faster growth in the future. This scenario has motivated many specialists to take advantage of the observed past behavior of shares as well as financial indexes to help avoid periods of instability within a company, or even better, to predict its behavior in the near future, allowing them to advise investors about possible risks. Stock markets generate large amounts of data concerning companies shares. The continuous data flow is part of sets of historic prices and variations of the value of a company. According to many authors [8,11], it is possible to use this data to predict a good opportunity for trading and use this prediction for investment, although accurate predictions are very hard to make due to the "chaotic" nature of stock markets, which depend on external events, such as governmental actions, company crises, natural disasters and many other factors. It can be found several attempts to model economic time series without considering outside events were made in the last decades, analyzing only the properties of the long and short-term history of the time series [13].

The objective of this paper is to identify signals that help to predict where prices may move in the future in the shortest possible time, based on similarity concepts. To do so, we propose a new forecasting content-based method that obtains relevant features from financial time-series based on the intrinsic information existing inside stock market time series and enables the creation of classification models based on their similarity and employing Metric Access Methods to speedup the process. We propose a new feature extractor for financial time series, which fragments the series using user-defined windows and extracts intrinsic features that are visually intuitive for the analysts. The forecasting is done through an Instance-Based Learning classifier [1] based on the dissimilarity between series fragments. Our method allows us to accurately answer questions such as the following: *"Regarding the Company Y shares, their values for Monday, Tuesday, Wednesday, Thursday are known. The Stock Market opens on Friday having Company Y shares with value X. Should we invest in Y expecting that the share will increase more than 1%?"*. Here we propose employing our method to perform a classification based on the nearest neighbor operation through an *IBk* classifier to decide whether or not it is worth investing in Y. To evaluate our method, we employed two real financial time series datasets to compare the accuracy and the speed of our method to that of the ARIMA and ANN-MLP methods. Through the Wilcoxon discrete test [18] we show, with a high level of confidence, that our approach was more effective than the other methods as well as more efficient.

The rest of this paper is structured as follows. Section 2 reviews the related work and presents the background necessary to follow the paper. Section 3 describes the proposed method. Section 4 shows the experimental results obtained comparing our approach to related methods, and Section 5 presents the conclusions.

2 Background and Related Work

2.1 Methods for Financial Time Series Analysis and Forecasting

In stock market analysis, the behavior of share pricing is usually described by discrete-time series, where each item of the series is a set of attributes regarding the price variations. Formally, a time series is defined as follows.

Definition 1 (Time series). *A time series is a set of observations x_t, each one being recorded at a specified time t.*

Time series is a complex data type composed by multiple data items, which can be represented in several ways. Considering that each observation x_t is the value of a certain random variable X_t, the time series is a particular realization $\{x_t\}_1^T$ of a stochastic process, which is the family of random variables $\{X_t\}_{-\infty}^{\infty}$ defined on an appropriate probability space. Using this representation and following a set of conditions given by the Wold decomposition theorem, it is also possible to represent the time series as a linear combination of a sequence of uncorrelated random variables [13].

The representation of time series as an univariate linear stochastic process is the development basis of the Autoregressive Model (AR) and the Moving Average Model (MA), both members of the Autoregressive Integrated Moving Average (ARIMA) model [5], which has been one of the most popular approaches to time series forecasting. The ARIMA model takes advantage of both weak-stationary and non-stationary properties of time series and proposes to integrate auto regressive and moving averages, offering ways to identify seasonalities observed at the series micro-level. It is based on three parameters: lags of difference (the auto regressive term), the difference order term and the lags of forecast errors (the moving average term). An important point to forecast economic time series using ARIMA is the right choice of the parameters that determine the model. ARIMA has been employed to assess a wide range of problems regarding financial time series, such as currency exchange and share behavior on the stock market [13]. When non-linearities are present in the data, the linear stochastic models fail to predict aspects of interest in series. Non-linear methods to model share price behavior include the Autoregressive Conditional Heteroskedasticity (ARCH) method [4] and its variations, that estimate non-constant values for the volatility, models based on the Hidden Markov Chains (HMM) [9], and the Method of Analogues [7]. These models have some drawbacks, such as the lack of scalability of the HMM technique and the limitation of the Method of Analogues to handle long periods of time and high-dimensionality series [6].

Soft-computing approaches based on artificial intelligence methods have suggested new ways to forecast time series outcomes [7]. These methods usually aim to predict trends using a classifier that gives an interpretation of summarized data. Among these approaches, Artificial Neural Networks (ANN) have been one of the most popular tools used in recent works regarding the financial market [12]. The numeric nature of ANN avoids data conversion problems faced by symbolic manipulation techniques and there is no need to any data distribution assumption for input data. Moreover, ANN have been successfully combined

with traditional statistic approaches, yielding relevant improvements [10]. However, ANN approaches require long training times [6]. Other widely adopted soft-computing approach for time series forecast is Support Vector Machines (SVM). Several works have been employing SVM and Support Vector Regression for financial series forecasting [2,17]. Nevertheless, the inherent "black-box" processing of ANN and SVM does not provide intuitive clues for the analyst.

A different category of works aims at performing financial time series forecasting based on the similarity of the current situation (i.e. a piece of the series) with past ones. This category includes approaches based on Case-Based Reasoning (CBR) [14], which consists in solving new problems by adopting solutions to analogous problems, and approaches guided by complex data similarity search and classification [15]. These methods require to extract relevant time series features to be employed as a signature for the (pieces of) original series data and to define an adequate measure to compute similarity between such signatures. The features must describe characteristics that are intelligible by the market analyst to allow a clearer interpretation and to improve his/her confidence on the system's solution. The solution we propose in this paper follows the complex data similarity search and classification approach. Our work differs from existing ones as it is designed to be a database-centric solution integrated to a SQL-compliant engine and it focus the financial time series similarity based on visual interpretations of data that are habitually employed by market analysts. Therefore, our proposal can not only be used to perform market forecasting, as we show in the experiments section, but also provide consistent and scalable primitives to develop advanced analysis tools over it. The following sections introduce the main concepts to understand our work.

2.2 Similarity Retrieval and Instance-Based Learning Classification

Complex data, such as multimedia data and time series, enclose rich information that demand specialized query operations for providing effective retrieval. In this context, similarity queries are among the most widely employed operations. Similarity queries retrieve the stored elements that satisfy a given similarity criterion regarding one or more query elements. Complex data elements have their intrinsic information extracted into feature vectors that are used in place of the original (raw) data in the query execution. The similarity evaluation usually relies on distance functions that quantifies how close two elements are in the feature space. The closer the elements are the higher is the similarity between them. Are of particular interest the distance functions that allow defining metric spaces as metric spaces allow using the indexing structures known as Metric Access Methods (MAM) to speedup query execution [19]. A MAM employs the properties of the underlying metric to prune elements, especially the triangular inequality. One example is the Slim-Tree, which is used by many systems and is able to answer similarity queries more than two hundred times faster than using a sequential scan [16]. There are several metrics in the literature. The functions of the Minkowski family (L_p) are the most widely adopted by similarity retrieval systems.

The main types of similarity queries are the Range query (R_q) and the k-Nearest Neighbor query (k-NNq). A Range query retrieves the elements in the dataset that are not farther from the query element than a given dissimilarity threshold, while the k-Nearest Neighbor query retrieves a quantity of similar elements independent of the threshold. These queries are useful in many situations when analyzing a financial time series. For example, a typical query in a stock market decision support tool is: *"Find the* 3 *share weeks in the economic time series whose behaviors are the most similar to the behavior of the current week, with regard to a given (set of) company(ies)"*. There are systems that support similarity queries. One example is the SIREN (*Similarity Retrieval Engine*) [3], which is a prototype that implements a blade over a commercial DBMS that interprets an extended-SQL syntax that include similarity query operators.

Similarity can also be useful for time series forecasting. The general idea is to provide a prediction based on the behavior of a previous similar situation, which is recorded in any part of the historical series, assuming that situations with similar behaviors have similar outcomes. This procedure usually relies on an Instance-Based Learning (IBL) classifier, which is a classifier derived from the k-NN query. The simplest IBL algorithms is the the *IB1*. The *IB1* algorithm sets the class of the target element as that of its closest neighbor in the concept description, using the Euclidean spatial distance to measure the dissimilarity between instances and adopting a simple policy for tolerating missing values in the feature vectors. Variations of the *IB1* include the *IB2* algorithm, which is identical to *IB1* except that it only saves misclassified instances, and the *IB3* that employs a "wait and see" method to determine which of the saved instances are expected to be helpful for future classifications [1]. Instance-Based Learning classifiers have several advantages. First of all, their approach is simple and follows the natural experience-based human reasoning, which allows supporting rigorous analyses in a way that complies with the analyst intuition. The IBL algorithms are also relatively robust, as they tolerate noisy and irrelevant attributes (although it is not the case here), and they can represent both probabilistic and overlapping concepts. Finally, as the IBL classification is based on k-NN queries, it can be homogenously integrated with other content-based retrieval operations in a framework for time series analysis and forecasting, which is the focus of this paper.

2.3 Statistical Measurement of the Classification Effectiveness

When it is desired to compare the success rate of two classifiers, in order to define the best one for a given application, a good alternative is to use a statistical hypothesis test. Intuitively, it is wanted to compare the difference between the success rate of the classification reported by each method. This is the main idea of the hypothesis test proposed by Wilcoxon [18]. The signed-rank test of Wilcoxon is a non-parametric alternative that can deal with discrete variables. In the context of this paper, every difference in the performance of two classifiers for each dataset, ignoring the signs, receives a number, or a "rank" in the list. Formally, it can be written as two hypotheses: the null hypothesis H_0 that states

that the classifier A is equal to B, regarding the classification performance, and the the hypothesis H_1 that states that classifier B is better than A. The test is done as follows. Let d_i be the difference between the performance score of the classifier B and of the classifier A on the i^{th} of the n elements in the test dataset. Let $R+$ be the sum of ranks in which B outperformed A and let $R-$ be the sum of ranks in which the opposite happened, defined by equations following.

$$a)\ R+ = \sum_{d_i>0} rank(i) + \frac{1}{2}\sum_{d_i=0} rank(i) \quad b)\ R- = \sum_{d_i<0} rank(i) + \frac{1}{2}\sum_{d_i=0} rank(i)$$

where the ranks are defined according to the absolute value of the differences d_i and ties ($d_i = 0$) imply half score for $R+$ and half for $R-$. Let T be the smaller of the sums, $T = \min\{R+, R-\}$. Most books on statistics include a table of exactly critical values for T for n up to 25 or so. However, the approach of Wilcoxon can be approximated by the Standard Normal Distribution for a larger number of samples, as shown in Equation 1. Fixing a confidence level α, the z value can easily be checked in a table for z-values. This allows concluding whether the null hypothesis should be accepted or rejected.

$$z = \frac{T - \frac{1}{4}n(n+1)}{\sqrt{\frac{1}{24}n(n+1)(2n+1)}} \tag{1}$$

3 The Proposed Approach

The approach we propose in this paper integrates similarity search and classification techniques in a common framework to provide financial time series analysis and forecasting primitives through an efficient database-driven architecture. The main components of our architecture are: financial time series feature extractors; distance functions; indexing structures for similarity search; instance-based classifiers; and an extended-SQL interface on top of a DBMS for end-user application connections. This allows answering efficiently several questions, such as:

- What are the companies that usually have an income greater than 3% for two consecutive months?
- Find the five weeks in the historical series that are the most similar to the current one.
- We are expecting a profit of 3%; should we buy shares of company Y in August/2013 to sell in September/2013?

Answering these questions uses distinct computational techniques provided by the proposed architecture. The first question employs conventional queries, that are natively provided by the underlying DBMS, while the second one uses the similarity search engine over a financial time series and the third relies on an instance-based classification algorithm attached to the engine.

Our approach follows the process illustrated in Fig. 1. Every step of the process is performed automatically, which not only avoids human subjectivity but

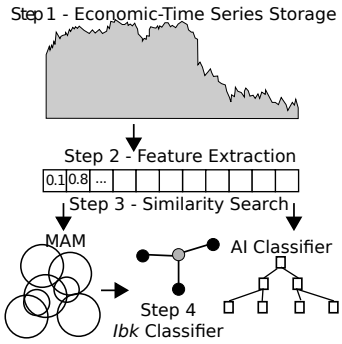


Fig. 1. The proposed process to financial time series analysis and forecasting

also provides an explainable technique to determine the best combination of each classification or analysis goal. The **first step** is the time series storage. The time series is represented in our approach as a table in which each tuple is an observation. Therefore, it can be stored several different series, according to the applications' requirements. The **second step** of our approach process is the feature extraction. This procedure is composed by two parts. The first part is the fragmentation of the time series in pieces according to a user-defined window. The window size (i.e. the number of composing observations) as well as the observations' granularity can vary, as they are application-dependent. Changing the observations' granularity is done manually by the user through DBMS aggregation function queries, while the window size is a parameter provided to the feature extractor. Subsequent time series fragments can overlap, as this characteristic is useful for some applications. This behavior is controlled by another parameter provided to the extractor, called window offset, which indicates the number of observations that the algorithm must go forward from the begining of a fragment to the begining of the next fragment. Each fragment is automatically saved as a tuple in an auxiliar table controlled by the system, converting the fragment observations into column values. The second part of the feature extraction process is the extraction of the intrinsic characteristics of the fragments. The resulting feature vector is stored as an additional column in the auxiliar table. The **third step** of Fig. 1 is the similarity search. This step allows posing several types of similarity queries, using different distance functions, such as the Manhattan, Euclidean and Chebychev Minkwosky functions (Section 2.2), and executing the queries over a Metric Access Method. These functionalities are provided by the SIREN in which we implemented the time series analysis and forecasting methods. The **last step** of the process is the classification process that provides the forecasting capabilities of the approach. It is available the *IBk* algorithm (an *IB1* variation whose classification is given analyzing the k nearest neighbors), however other instance-based learning classifiers can be easily included. Every new functionality provided by our approach is included in the SIREN through SQL user-defined functions and extended-SQL instructions. Therefore,

any application that connects to SIREN can benefit from the time series analysis and forecasting primitives to build financial decision support applications.

3.1 A New Feature Extractor for Stock Market Time Series

The success of the similarity evaluation is directly linked to the intrinsic characteristics extracted to represent the data. Thus, the more accurate the features are the closer the distances between the target and the similar instances and the better the classification. In this section we present a new feature extractor aimed at analyzing and predicting the shares' behavior on the stock market. This extractor takes advantages of the relation between the price variations of a share during a period, usually a day. To accomplish the requirement that our approach should be intuitive and understandable, in contrast to "black-box" methods, the extractor is based on candlesticks, defined as follows.

Definition 2 (Candlestick). *A candlestick is a visual representation for a subset of the information represented by each observation in a stock market series.*

Candlesticks are the most common data representation for market analysts. Fig. 2 (A) shows the visual representation of the two types of candlesticks: the red candlestick, which represents a time series observation (usually covering a day) with negative closing, and the green candlestick that corresponds to a day with positive closing value [11]. Each candlestick has four points: the opening price (the share's price at the beginning of the day on the stock exchange), the closing price (the price at the end of the day), the high price (the maximum price during the day) and the low price (the share's minimum price).

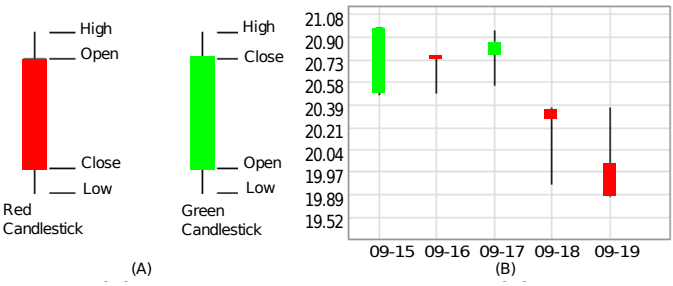


Fig. 2. (A) Candlestick representations for economic time series. (B) A candlestick representation for a share's week starting on 09-15.

The chart analysis offers some visual features that describe the behavior of the share, which can be extracted without losing their semantic meaning. For example, Fig. 2 (B) shows a plot graph of a given share based on candlesticks for the week starting on 09-15. The day 09-16 shows a graphical phenomenon

known as a "nail" and the day 09-19 shows a graphical phenomenon known as a "hammer", which are phenomenon very familiar to market analysts. The candlesticks provide obvious graphic distinctions on the correlation between the average of the opening and closing points of each day, as well as the variance of the maximum and minimum values of the share on a given day. The closing price of the share is not necessarily its opening price on the next day, generating gaps between consecutive days. That gaps are very important for analyzing the stock market time series because it describes the external influence (for example, political and environmental events that can affect the trust of the investor). If the gap is small, it could be a trend towards the recovery of the share price in the future. Oscillation periods due to external factors can also be better represented by this feature, improving future classifications that could relate similar periods. The average behavior projected on the amplitude of the opening-closing relation can be better described by the variation of the share price during a day, describing stable or unstable share behavior.

The feature extractor proposed in this paper considers only features expressed in candlesticks (intrinsic information of the share movement). We do not consider other external variables, like the trending volume, although they could be used for weighting particular features in the distance function. The features extracted are the following.

1. **Body-Shadow Relation** – The relation body-shadow is a linear function expressed by the daily values represented on the candlestick (whose visual interpretation is shown in Fig. 3 (A)) as:

$$f(x) = \begin{cases} \frac{\left(open(x)+close(x)\right)/2-low(x)}{high(x)-low(x)} & \text{if } high(x) - low(x) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

2. **Amplitude Relation** – The amplitude relation is a linear function expressed by the daily values represented by the candlestick (whose visual interpretation is shown in Fig. 3 (B)). This feature describes phenomenons such as the "nail" and "hammer" and is given by:

$$f(x) = \begin{cases} \frac{open(x)-close(x)}{high(x)-low(x)} & \text{if } high(x) - low(x) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

3. **Gap Relation** – The gap relation is a linear function between two consecutive days in the economic time series. This relation captures the variations of non-continuous price descriptions, and it can be viewed in Fig. 3 (C). The closing prices are always greater than 0 for active companies.

$$f(x) = \frac{open(x) - close(x-1)}{close(x-1)}. \tag{2}$$

4. **Projection Relation** – The projection relation is a function that describes the average of the amplitude projected over the day average, as shown in Fig. 3 (D).

$$f(x) = \begin{cases} f(x) = \frac{\left(open(x)+close(x)\right)/2-low(x)}{open(x)-close(x)} & \text{if } open(x) - close(x) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

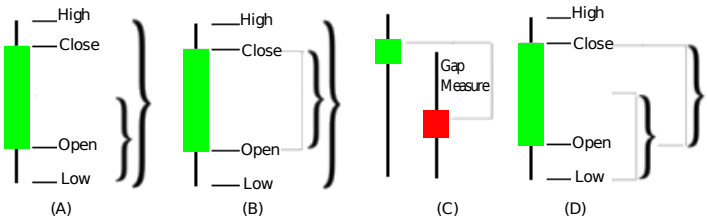


Fig. 3. Visual representation of the proposed features: (A) Body-Shadow Relation (B) Amplitude Relation (C) Gap Relation and (D) Projection Relation

The minimum window size is two because of the Gap Relation feature. For a better performance, when compared with distance functions, such as the Euclidean function, it is also required to perform a feature set normalization.

4 Experimental Results

In this section we show experimental results about employing the proposed technique to classify and to index financial time series. The experiments presented aimed at testing: (1) if our proposed technique forecasts more accurately than the methods ARIMA and ANN-MLP (Multi-Layer Perceptron ANN); (2) if our technique is faster and more scalable than the aforementioned methods. The experiments were run over the implementations done on the SIREN prototype.

We used two real datasets for the experiments. The first one, hereafter referred to as BM&FBOVESPA_SHARES, is composed by time series regarding the 33 largest shares in the BM&FBOVESPA stock exchange, according to their average volume of daily transactions in 2010. The period of the dataset starts in 01-01-2005 and ends in 01-01-2010. Yahoo Finances[1] provides a free database and analysis for these companies. The main motivation for considering this scenario is the significant growth of investors in the Brazilian market over the past five years [12] and the lack of studies on this stock exchange [2]. The second dataset, referred to as WORLD_STOCKINDEXES, is composed by series regarding the 7 largest stock exchanges indexes around the world: Dow Jones (New York), FTSE (London), BM&FBOVESPA (São Paulo), Nikkei (Tokyo), SSMI (Bern), Euronext (Paris) and SSEC (Shanghai). Each series has a particular start but all of them end in 01-01-2010.

[1] `http://finance.yahoo.com`

4.1 Forecasting Accuracy

To evaluate the forecasting accuracy of the methods, we first tested the accuracy to forecast stock market indexes, as they describe the behavior of an entire stock exchange series and analyzing them is very important for the investors to gain confidence. This experiment used the WORLD_STOCKINDEXES dataset to forecast the Friday high index values concerning the earlier week days. Table 1 shows the hit ratio (percentage) achieved by our method, ARIMA and ANN-MLP, with regard to each of the seven indexes. It can be noticed that our method outperformed the ARIMA method regarding every index and that our method also outperformed the ANN-MLP, with the exception of the ^SSMI index.

We also compared the methods regarding individual shares. To do so, we performed an experiment whose goal was to answer the question posed at the beginning of the paper: *"Regarding the Company Y shares, their values for Monday, Tuesday, Wednesday, Thursday are known. The Stock Market opens on Friday having Company Y shares with value X. Should we invest in Y expecting that the share will increase more than 1%?"*. Fig. 4 illustrates the meaning of this query using real instances of the dataset. Fig. 4 (A) shows the stock behavior of the week whose Friday's outcome is wanted to predict. Figures 4 (B), 4 (C) and 4 (D) show, respectively, the 1^{st}, 2^{nd} and 3^{rd} nearest neighbors of the query week, regarding the first 4 days. The classification done by the *IBk* is the class of the majority of the nearest neighbors returned, which matches the actual Friday's result in this example (Fig. 4 (E)).

Table 2 shows the prediction accuracy achieved by the tested methods regarding every company of the BM&FBOVESPA_SHARES dataset. In the table, the columns 2 to 4 show the hit ratio of the methods (percentage), the ZeroR column shows the class distribution, the values $|d_{ij}|$ and $rank_{ij}$ are the values obtained for the Wilcoxon test between our method and ARIMA, and the values $|d_{ik}|$ and $rank_{ik}$ are the needed values for the test between our method and ANN-MLP. The following comparisons are based on these values.

Table 1. Methods' accuracy regarding the WORLD_STOCKINDEXES dataset

Index	Acronym	Our Method %	ARIMA %	ANN-MLP %	ZeroR %
Dow Jones I A	^DIJ	81.76	68.27	78.54	50.76
London - FTSE	^FTSE	79.10	66.23	79.00	51.89
Tokyo	^NIKKEI	75.91	66.17	73.23	54.17
Swiss Market	^SSMI	71.23	62.25	72.37	50.42
Euronext Paris	^FHCI	70.38	66.35	70.05	52.25
BM&FBovespa	^BVSP	63.61	57.17	60.23	52.16
Shangai S E	^SSEC	61.63	56.32	59.47	51.18

Our Method vs. ARIMA. To compare our predictions with those of ARIMA, we applied the Wilcoxon Hypoyhesis test. Let A be the accuracy rate achieved by the ARIMA model and let B be the accuracy rate achieved by our method. In that case, we consider the following hypothesis:

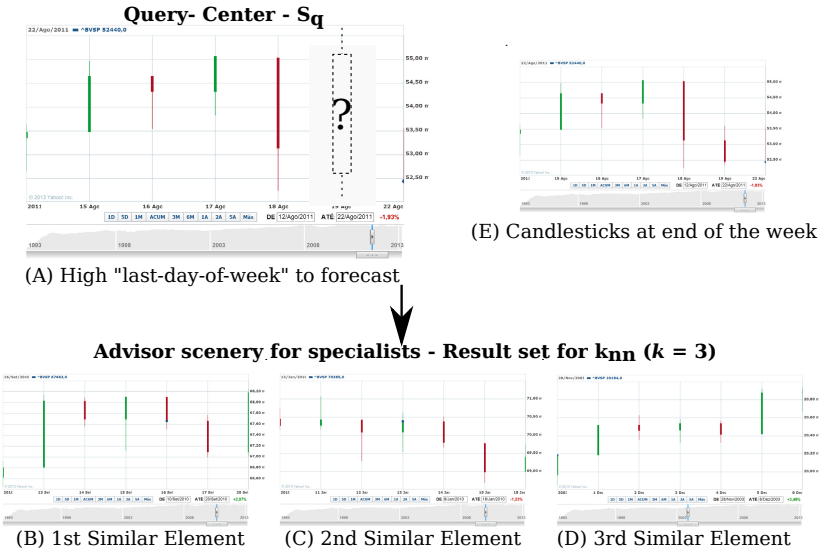


Fig. 4. Our similarity approach: given a query center (A), the expert can visualize the most similar financial time series fragments (in this case $k = 3$ and (B), (C) and (D) are the 3 most similar elements) or demand the forecast over it. The image (E) shows the actual behavior of the query center at the end of Friday, which matches the *IBk* classification in this example.

1. Let $H_0 : A = B$ be the null hypothesis, which means that using either method A or method B to perform forecasts produces similar results;
2. Let $H_1 : A < B$ be the alternative hypothesis, which means that using method B is more efficient to perform forecasts than using method A.

Let $\alpha = 0.1$. The values of $R+$ and $R-$ are 438 and 123, respectively. The minimum value for T is 123, thus $z = -2.793414$. According to the Normal Distribution Table, $P(X < -2.793414) < 0.05$. Thus, we must reject H_0 and accept H_1, which means that using our method is more accurate than using the ARIMA model to perform forecasting.

Our Method vs. ANN-MLP. Let B be the accuracy rate achieved by our method, and let C be the accuracy rate achieved by the ANN-MLP classifier. Then consider the following hypothesis:

1. Let $H_0 : C = B$ be the null hypothesis, which means that using either method C or B to perform forecasts produces similar results;
2. Let $H_1 : C < B$ be the alternative hypothesis, which means that using the method B is more efficient than using method C to perform forecasting.

Consider a confidence level $\alpha = 0.1$. The next step is to calculate $R+$ and $R-$, based in $rank_{ik}$. According to Table 2, we have: $R+ = 375$ and $R- = 186$. The

Table 2. Methods' accuracy regarding the BM&FBOVESPA_SHARES dataset

Share	Our Method %	ARIMA %	ANN-MLP %	ZeroR %	$\|d_{ij}\|$	$\|d_{ik}\|$	$rank_{ij}$	$rank_{ik}$
BOBR4	65.7038	57.00	66.6823	51.2077	8.6038	0.9785	32	10
CRUZ3	65.0713	61.50	62.8940	56.0386	3.5713	2.1772	16	25
AMBV4	64.495	57.86	62.5633	52.0202	6.635	1.932	30	24
PETR4	64.9155	60.57	62.2290	54.3269	4.3455	2.6865	20	29
BRAP4	62.6696	58.55	61.0671	50.2703	4.1196	1.6025	19	19
CCRO3	57.5423	61.27	57.0450	52.973	3.7277	0.4973	17	5
PSSA3	56.5169	48.10	58.0987	53.1401	8.4169	1.5818	31	18
EMBR3	60.6664	54.33	61.3840	50	6.3364	0.7176	28	7
VALE5	58.7958	53.55	58.5430	55.5556	5.2458	0.2528	24	3
TMAR5	56.3266	56.31	57.6486	50.7246	0.0166	1.3220	1	17
ELET6	63.6079	58.98	62.4890	56.5217	4.6279	1.1189	22	13
BBDC4	56.7801	55.81	58.6534	51.3889	0.9701	1.8733	4	23
TNLP4	55.3423	53.87	55.2211	57.971	1.4723	0.1212	6	2
USIM5	59.5722	53.14	57.1590	50.9615	6.4322	2.4132	29	26
TLPP4	61.3458	56.21	58.5647	51.3514	5.1358	2.7811	23	31
CMIG4	60.7181	61.45	58.9773	53.1401	0.7319	1.7408	3	20
LAME4	56.8193	54.28	57.9146	50.2415	2.5393	1.0953	10	12
TAMM4	61.1337	55.38	62.3780	56.0386	5.7537	1.2443	26	15
ITSA4	55.8233	55.26	55.2455	51.3514	0.2633	0.5778	2	6
ELPL4	60.8649	58.21	58.4333	51.0067	2.6549	2.4316	11	27
LIGT3	56.9298	54.87	54.2333	51.2077	2.0598	2.6965	8	30
GOLL4	58.2230	54.37	57.3221	50.8108	3.853	0.9009	18	9
DASA3	59.1217	56.45	59.0145	55.157	2.6717	0.1072	12	1
NATU3	55.7391	54.46	53.9903	52.4324	1.2791	1.7488	5	21
CSNA3	52.479	49.07	54.3245	50.7246	3.409	1.8455	14	22
BBAS4	60.7447	55.45	59.8670	54.29	5.2947	0.8687	25	8
LREN3	50	54.47	52.6793	51.6908	4.47	2.6793	21	28
PDGR3	55.9602	53.46	56.2560	55.2632	2.5002	0.2858	9	4
GGBR4	57.996	55.25	54.9099	59.9099	2.746	3.0861	13	32
RSID3	46.1132	57.86	47.2890	56.5217	11.7468	1.1758	33	14
BRKM5	57.656	59.31	56.5678	51.6584	1.654	1.0882	7	11
BRFS4	49.3213	52.76	44.2595	61.7117	3.4387	5.0617	15	33
EBTP3	47.2426	53.55	48.5002	63.1068	6.3074	1.2576	27	16

T value is 186. According to the Standard Normal Distribution, we should reject the null hypothesis if $P(X < z) \leq \alpha$, where $z = -1.67604836$. Following the Normal Distribution Table, we have $P(X < -1.67604836) < 0.1$. Thus, we must reject H_0 and accept H_1, which means that our method is more accurate than using the proposed features associated with an ANN-MLP classifier.

4.2 Speed and Scalability Analysis

Besides the accuracy, the execution time demanded by the analyses is another important point to consider. Therefore, we analyzed the computational effort demanded for executing the stock market index forecasting using the dataset WORLD_STOCKINDEXES. In these tests we evaluated two versions of our approach: one that does not use an access method to accelerate the classification (*IBk*) and another that uses a Slim-tree to do so (Boosted *IBk*). In this section,

we show the results achieved using the Dow Jones index (^DIJ), which is the largest series we tested.

Fig. 5 (A) shows the total time demanded by the methods to perform the stock market index forecasting experiment (Friday high values) regarding the ^DIJ series, varying the series size. The ANN-MLP was the fastest method, presenting a quasi-constant behavior. The Boosted *IBk* was the second fastest method, being up to 130% faster than the traditional *IBk* and up to 720% faster than ARIMA. It is worth noticing that this plot considers that the access method used by Boosted *IBk* is already created and that the ANN-MLP is trained. Creating the Boosted *IBk* Slim-tree for the ^DIJ series took about 190 seconds on average and training an ANN-MLP took more than 360 seconds on average. If the setup time is also considered, the ARIMA was the fastest method, however the least accurate, our method was the second fastest and the most accurate and the ANN-MLP was the slowest. The speedup achieved by the Boosted *IBk* over the traditional *IBk* was a result of the reduction in the number of distance calculations (up to 20%), presented in Fig. 5 (B), and in the number of disk accesses (up to 70%), showed in Fig. 5 (C).

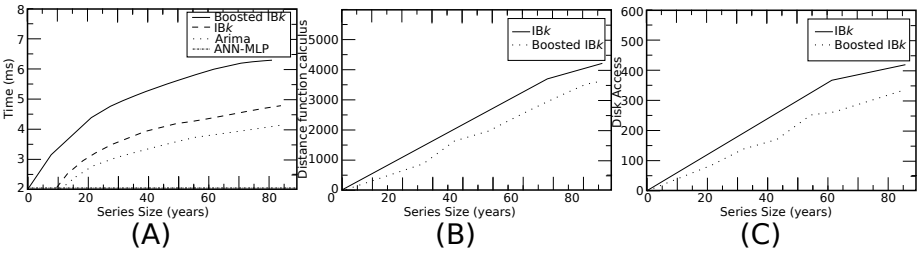


Fig. 5. Forecast speed comparison among the methods. (A) Total time. (B) Number of distance calculations. (C) Number of disk accesses.

5 Conclusions

Forecasting financial time series is difficult because the data is very noisy and chaotic. The predictions are even harder when trying to forecast shares in the markets of developing countries, which are subject to more expressive unforeseen external changes. This paper presented a new nonlinear forecasting and analysis approach using a "white-box" processing. The proposed approach allows identifying correlations between financial time series fragments using a graphical representation and perform fast classification through the integration the of an *IBk* classifier and a MAM indexing the extracted features. The experiments showed that our technique can perform forecasting more accurately and faster than the other tested methods. Moreover, our work, as a time series indexing technique [10], can also be used for similarity searches and forecasting on very large time series datasets.

References

1. Aha, D., Kibler, D.: Instance-based learning algorithms. Machine Learning 6, 37–66 (1991)
2. Atsalakis, G.S., Valavanis, K.P.: Surveying stock market forecasting techniques. Expert Systems with Applications 36(3, Part 2), 5932–5941 (2009)
3. Barioni, M.C., Razente, H., Traina, A., Traina Jr., C.: SIREN: a similarity retrieval engine for complex data. In: VLDB, pp. 1155–1158 (2006)
4. Bollerslev, T., Ray, Y.C., Kroner, K.F.: Arch modeling in finance: A review of theory and empirical evidence. Journal of Econometrics 52(1-2), 5–59 (1992)
5. Box, G.E.P., Jenkins, G.: Time Series Analysis, Forecasting and Control. Holden-Day, Incorporated (1990)
6. Chakrabarti, D., Faloutsos, C.: F4: large-scale automated forecasting using fractals. In: CIKM, pp. 2–9. ACM (2002)
7. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: SIGMOD, pp. 419–429. ACM (1994)
8. Granger, C.W.J.: Forecasting stock market prices: Lessons for forecasters. International Journal of Forecasting 8(1), 3–13 (1992)
9. Hassan, R., Nath, B.: Stockmarket forecasting using hidden markov model. In: ISDA, pp. 192–196. IEEE Computer Society (2005)
10. Huang, C.-J., Yang, D., Chuang, Y.: Application of wrapper approach to the stock trend prediction. Expert Systems with Applications 34, 2870–2878 (2008)
11. Lee, K.H., Jo, G.S.: Expert system for predicting stock market timing using a candlestick chart. Expert Systems with Applications 16(4), 357–364 (1999)
12. Martinez, L.C., da Hora, D.N., de, J.R., Palotti, M., Meira, W., Pappa, G.L.: From an artificial neural network to a stock market day-trading system: a case study on the bm&f bovespa. In: IJCNN, pp. 3251–3258. IEEE Press (2009)
13. Mills, T.C., Markellos, R.N.: The econometric modelling of financial time series, 3rd edn. Cambridge Univ. Press, Cambridge (2008)
14. Simić, D., Simić, S., Svirčević, V.: Invoicing and financial forecasting of time and amount of corresponding cash inflow. Management Information Systems 6(3), 14–21 (2011)
15. Suntinger, M.: Event-based similarity search and its applications in business analytics. Master's thesis, Vienna University of Technology, Vienna, Austria (2009)
16. Traina Jr., C., Traina, A., Faloutsos, C., Seeger, B.: Fast indexing and visualization of metric data sets using slim-trees. IEEE TKDE 14, 244–260 (2002)
17. Wang, B., Huang, H., Wang, X.: A novel text mining approach to financial time series forecasting. Neurocomputing 83(0), 136–145 (2012)
18. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945)
19. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search - The Metric Space Approach, vol. 32. Springer (2006)

Inferring Knowledge from Concise Representations of Both Frequent and Rare Jaccard Itemsets

Souad Bouasker[1] and Sadok Ben Yahia[1,2]

[1] LIPAH, Computer Science Department, Faculty of Sciences of Tunis, Tunis, Tunisia
[2] Institut TELECOM, TELECOM SudParis, UMR 5157 CNRS SAMOVAR, France

Abstract. Correlated pattern mining has become increasingly an important task in data mining and knowledge discovery. Recently, concise exact representations dedicated for frequent correlated and for rare correlated patterns according to the *Jaccard* measure were presented. In this paper, we offer a new method of inferring new knowledge from the introduced concise representations. A new generic approach, called GMJP, allowing the extraction of the sets of frequent correlated patterns, of rare correlated patterns and their associated concise representations is introduced. Pieces of new knowledge in the form of associations rules can be either exact or approximate. We also illustrate the efficiency of our approach over several data sets and we prove that *Jaccard*-based classification rules have very encouraging results.

Keywords: Concise representation, Monotonicity, Constraint, Correlated pattern, *Jaccard* measure, Generic Approach.

1 Introduction and Motivations

Correlated item set mining is at the core of numerous data mining tasks. The enormous research efforts dedicated to this topic have led to a variety of sophisticated approaches [2,8,11,20,21]. In this regard, a variety of correlation measures were proposed and studied. In this work, we will focus on the *Jaccard* correlation measure [9]. Indeed, the *Jaccard* measure was used in many works under various names like *coherence* [14], *Tanimoto coefficient* [24] and *bond* measure [18] [(1)]. The *bond* measure was recently redefined in [3], where a concise exact representation of the set of frequent correlated patterns according to the *bond* measure was also proposed. Moreover, a generic approach for frequent *Jaccard* patterns mining was also performed in [20].

Frequent correlated itemset mining was then shown to be an interesting task in data mining. Since its inception, this key task grasped the interest of many researchers since it meets the needs of experts in several application fields [3], such as market basket study. However, the application of correlated frequent patterns is not an attractive solution for some other applications, *e.g.*, intrusion detection, analysis of the genetic confusion from biological data, pharmacovigilance, detection of rare diseases from medical data, to cite but a few [5,12,15,17,19,23]. As an illustration of the rare correlated patterns applications in the field of medicine, the rare combination of symptoms can provide useful

[1] In the rest of this paper, we used '*bond*' as a reference for the *Jaccard* measure.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 109–123, 2013.

insights for doctors [25]. We cite that in [6], the authors proposed a concise exact representation of the set of rare correlated patterns and designed a rule-based classification process.

It is to mention, that no previous approach allowing the extraction of both frequent and rare correlated patterns according to a specified correlation metric was proposed. To solve this challenging problem, we propose an efficient algorithmic framework, called GMJP, allowing the extraction of both frequent correlated patterns, rare correlated patterns and their associated concise representations. To achieve the genericity of GMJP, we distinguish four different running scenarios depending on the required output. We, also, design a rule-based classifier and we discover meaningful correlations in data for frequent itemsets as well as for rare ones.

The paper is organized as follows. Section 2 presents the background used throughout this work. We introduce in Section 3 the recently proposed concise exact representations of both frequent and rare correlated patterns according to the *Jaccard* measure. Section 4 reviews some related work. The generic proposed approach GMJP is detailed in Section 5. We report an empirical study on different datasets and an association-rules based classifiction process respectively in Sections 6 and 7. We conclude and skecth issues of future work in Section 8.

2 Preliminaries

We start by presenting the key notions related to our work. We first define a dataset.

Definition 1. (***Dataset***) *A dataset is a triplet* $\mathcal{D} = (\mathcal{T}, \mathcal{I}, \mathcal{R})$ *where* $\mathcal{T}$ *and* $\mathcal{I}$ *are, respectively, a finite set of transactions and items, and* $\mathcal{R} \subseteq \mathcal{T} \times \mathcal{I}$ *is a binary relation between the transaction set and the item set. A couple* $(t, i) \in \mathcal{R}$ *denotes that the transaction* $t \in \mathcal{T}$ *contains the item* $i \in \mathcal{I}$.

In this work, we are mainly interested in itemsets as a class of patterns. The two main kinds of support a pattern can have are defined as follows, for any non-empty pattern I:

- **Conjunctive support:** $Supp(\wedge I) = |\{t \in \mathcal{T} \mid (\forall\, i \in I, (t,i) \in \mathcal{R})\}|$
- **Disjunctive support:** $Supp(\vee I) = |\{t \in \mathcal{T} \mid (\exists\, i \in I, (t,i) \in \mathcal{R})\}|$

Table 1. An example of a dataset

	A	B	C	D	E
1	×		×	×	
2		×	×		×
3	×	×	×		×
4		×			×
5	×	×	×		×

Example 1. Let us consider the dataset given by Table 1. We have $Supp(\wedge \mathrm{AD}) = |\{1\}| = 1$ and $Supp(\vee \mathrm{AD}) = |\{1, 3, 5\}| = 3$.[2]

[2] We use a separator-free form for the sets, *e.g.*, AD stands for the set of items {A, D}.

An itemset $\mathtt{I}$ is frequent if its support Supp($\mathtt{I}$) is above a user-defined minimum support threshold *minsupp*, otherwise the itemset $\mathtt{I}$ is said to be infrequent or rare.

The constraint of rarity is monotone, *i.e.*, $\forall\ I,\ I_1 \subseteq \mathcal{I}$, if $I_1 \supseteq I$ and $Supp(\wedge I) < minsupp$, then $Supp(\wedge I_1) < minsupp$ since $Supp(\wedge I_1) \leq Supp(\wedge I)$. Thus, it induces an *order filter* [7] on the set of all the subsets of $\mathcal{I}$, $\mathcal{P}(\mathcal{I})$. Contrariwise, the frequency constraint induces an *order ideal* [7].

The *bond* measure [18] is mathematically equivalent to *Jaccard* [9]. It was redefined in [3] as:

$$bond(I) = \frac{Supp(\wedge I)}{Supp(\vee I)}$$

The set of correlated patterns associated to the *bond* measure is defined as follows.

Definition 2. (***Correlated patterns***) *Considering a minimum correlation threshold minbond, the set $\mathcal{CP}$ of correlated patterns is equal to: $\mathcal{CP} = \{I \subseteq \mathcal{I} \mid bond(I) \geq minbond\}$* [3].

The *bond* measure takes its values within the interval $[0, 1]$. While considering the universe of a pattern I [14], *i.e.*, the set of transactions containing a non empty subset of I, the *bond* measure represents the simultaneous occurrence rate of the items of the pattern I in its universe. Thus, the more the items of I are dependent on each other, (*i.e.* strongly correlated), the higher the value of the *bond* measure is, since $Supp(\wedge I)$ would be closer to $Supp(\vee I)$. We present in what follows the concise exact representations associated to correlated patterns.

3 Concise Exact Representations of Correlated Patterns

In [3] and in [6], the authors introduced concise exact representations of respectively frequent correlated and rare correlated patterns. The proposed approaches are based on the concept of correlated equivalence classes induced by the f_{bond} closure operator associated to the *bond* measure.

In each equivalence class, all the elements have the same f_{bond} closure and the same value of *bond*. The minimal patterns of a *bond* equivalence class are the smallest incomparable members, w.r.t. set inclusion and are called **Minimal correlated patterns**, while the closed pattern is the largest one and is called **Closed correlated patterns**. These two sets are defined [3] as follows:

Definition 3. (***Closed correlated patterns by*** f_{bond}) *The set $\mathcal{CCP}$ of closed correlated patterns by f_{bond} is equal to: $\mathcal{CCP} = \{I \in \mathcal{CP} \mid \nexists\ I_1 \supset I$: $bond(I) = bond(I_1)\}$.*

Definition 4. (***Minimal correlated patterns***) *The set $\mathcal{MCP}$ of minimal correlated patterns is equal to: $\mathcal{MCP} = \{I \in \mathcal{CP} \mid \nexists\ I_1 \subset I$: $bond(I) = bond(I_1)\}$.*

While integrating the frequency constraint with the correlation constraint, we can distinguish between two sets of correlated patterns, which are the "Frequent correlated

[3] We refer in the rest of the paper to the minimum support threshold by *minsupp* and to the minimum correlation threshold by *minbond*.

patterns" set and the "Rare correlated patterns" set. Now, based on these two previous sets, the concise exact representation of frequent correlated patterns was studied and proposed in [3], in addition to the concise exact representation of rare correlated patterns which was proposed in [6].

Definition 5. (***The set of frequent correlated patterns***) *[3] The set $\mathcal{FCP}$ of frequent correlated patterns is equal to:* $\mathcal{FCP} = \{I \subseteq \mathcal{I} \mid Supp(\wedge I) \geq minsupp \text{ and } bond(I) \geq minbond\}$.

Definition 6. (***Concise exact representation of the $\mathcal{FCP}$ set***) *[3] The representation $\mathcal{RFCCP}$ is based on the set of frequent closed correlated patterns:*

$$\mathcal{RFCCP} = \{(I, Supp(\wedge I), Supp(\vee I)) \mid I \in \mathcal{CCP} \text{ and } Supp(\wedge I) >= minsupp\}.$$

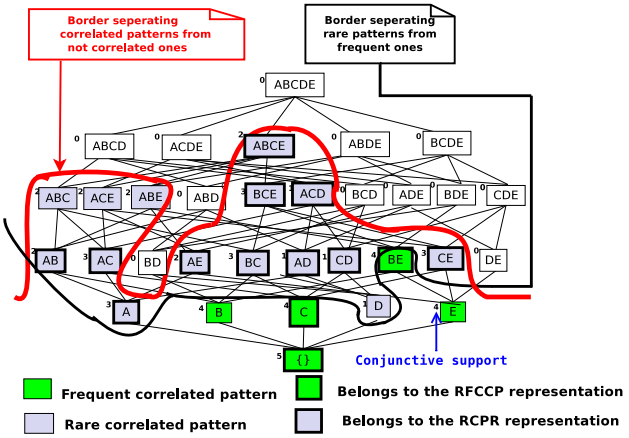


Fig. 1. Localization of the frequent correlated and the rare correlated patterns, and their associated condensed representation for *minsupp* = 4 and *minbond* = 0.2

Now, we present the rare correlated patterns associated to the *bond* measure:

Definition 7. (***The set of rare correlated patterns***) *[6] The $\mathcal{RCP}$ set of rare correlated patterns is equal to:* $\mathcal{RCP} = \{I \subseteq \mathcal{I} \mid Supp(\wedge I) < minsupp \text{ and } bond(I) \geq minbond\}$.

Definition 8. (***Concise exact representation of the $\mathcal{RCP}$ set***) *[6] Let $\mathcal{RCPR}$ be the concise exact representation of the $\mathcal{RCP}$ set based on the $\mathcal{CRCP}$ set of closed rare correlated patterns and on the $\mathcal{MRCP}$ set of the minimal rare correlated patterns. The $\mathcal{RCPR}$ representation is equal to:* $\mathcal{RCPR} = \mathcal{CRCP} \cup \mathcal{MRCP}$, *with*

$$\mathcal{CRCP} = \{(I, Supp(\wedge I), Supp(\vee I)) \mid I \in \mathcal{CCP} \text{ and } Supp(\wedge I) < minsupp\} \text{ and,}$$

$$\mathcal{MRCP} = \{(I, Supp(\wedge I), Supp(\vee I)) \mid I \in \mathcal{MCP} \text{ and } Supp(\wedge I) < minsupp\}.$$

These previous sets are depicted by Figure 1. The support shown at the top left of each frame represents the conjunctive support.

After presenting the *Jaccard* patterns, we propose in the next section, an overview of the approaches dealing with concise representations of correlated patterns.

4 Related Work

The problem of mining concise representations of correlated patterns was thoroughly studied in various works in the literature. The *bond* measure was studied in [13], the authors proposed an apriori-like algorithm for mining classification rules. Moreover, the authors in [20] proposed a generic approach for correlated patterns mining. Indeed, the *bond* correlation measure and eleven other correlation measures were used, all of them fulfill the anti-monotonicity property. Correlated patterns mining was then shown to be more complex and more informative than frequent patterns mining [20]. Also, in [22], a study of different properties of interesting measures was conducted in order to suggest a set of the most adequate properties to consider while mining rare associations rules. However, it is important to highlight that the extraction of rare correlated patterns was not carried out in [20] nor in [22].

Many other works have also emerged. In [26], the authors provide a unified definition of existing null-invariant correlation measures and propose the GAMINER approach allowing the extraction of frequent high correlated patterns according to the *Cosine* and to the *Kulczynski* measures. In this same context, the NICOMINER algorithm was also proposed in [10] and it allows the extraction of correlated patterns according to the *Cosine* measure. In this same context, we cite also the AETHERIS approach [21] which allow the extraction of condensed representation of correlated patterns according to user's preferences. In [2], the authors introduced the concept of flipping correlation patterns according to the *Kulczynsky* measure. However, the *Kulczynsky* does not fulfill the interesting anti-monotonic property as the *bond* measure. To the best of our knowledge, this work is the first one that puts the focus on mining concise representations of both frequent and rare correlated patterns according to the *bond* measure.

We introduce, in what follows, our new GMJP approach [4].

5 The GMJP Approach

We introduce in this section the GMJP approach which allows, according to the user's input parameters, the extraction of the desired output. As shown by Figure 2, four different scenarios are possible for running the GMJP approach:

- **First Scenario:** outputs the whole set $\mathcal{FCP}$ of frequent correlated patterns,
- **Second Scenario:** outputs the $\mathcal{RFCCP}$ concise exact representation of the $\mathcal{FCP}$ set,
- **Third Scenario:** outputs the whole set $\mathcal{RCP}$ of rare correlated patterns,
- **Fourth Scenario:** outputs the $\mathcal{RCPR}$ concise exact representation of the $\mathcal{RCP}$ set.

The GMJP algorithm takes as an input a dataset $\mathcal{D}$, a minimal support threshold *minsupp* and a minimal correlation threshold *minbond*. We mention that GMJP determines exactly the *support* and the *bond* values of each pattern of the desired output according to the user's parameters.

[4] GMJP stands for **G**eneric **M**ining of **J**accard **P**atterns.

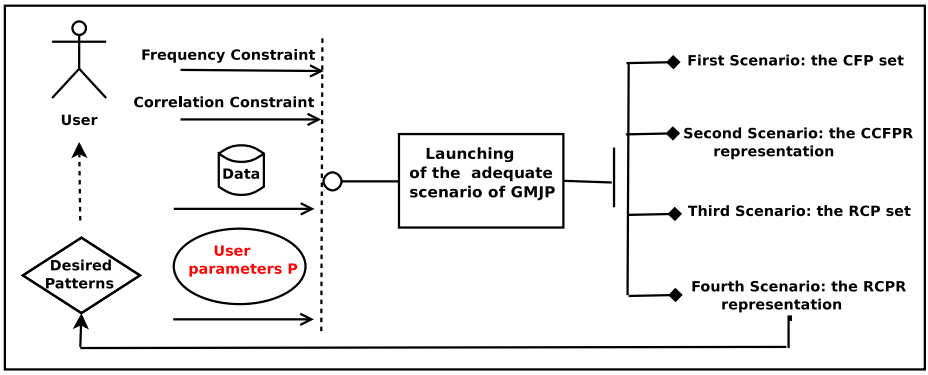


Fig. 2. Overview of GMJP

5.1 Overview of the Algorithm

We illustrate the different steps of GMJP when running the fourth script aiming to extract the $\mathcal{RCPR}$ representation. Our choice of this fourth scenario is motivated by the fact that the extraction of the $\mathcal{RCPR}$ representation corresponds to the most challenging mining task for GMJP.

In fact, $\mathcal{RCPR}$ is composed by the set of rare correlated patterns which results from the intersection of two theories [16] induced by the constraints of correlation and rarity. So, this set is neither an order ideal nor an order filter. Therefore, the localization of the elements of the $\mathcal{RCPR}$ representation is more difficult than the localization of theories corresponding to constraints of the same nature. Indeed, the conjunction of anti-monotonic constraints (*resp.* monotonic) is an anti-monotonic constraint (*resp.* monotonic) [4]. For example, the constraint "being a correlated frequent pattern" is anti-monotonic, since it results from the conjunction of two anti-monotonic constraints namely, "being a correlated pattern" and "being a frequent pattern". This constraint induces, then, an order ideal on the itemsets lattice. In fact, the GMJP algorithm mainly operates in three steps as depicted by Figure 3.

1. A first scan of the dataset is performed in order to extract all the items and assigning to each item the set of transactions in which it appears. Then, a second scan of the dataset is carried out in order to identify, for each item, the list of the co-occurrent items.
2. The second step consists in integrating both the constraints of rarity and of correlation in a mining process of $\mathcal{RCPR}$. In this situation, this problem is split into independent chunks since each item is treated separately. In fact, for each item, a set of candidates is generated. Once obtained, these candidates are pruned using the following pruning strategies:

   (***a***) **The pruning of the candidates which check the cross-support property [3].**
   (***b***) **The pruning based on the order ideal of the correlated patterns**.

   Recall that the set of correlated patterns induces an order ideal property. Therefore, each correlated candidate, having a non correlated subset, will be pruned since it will not be a correlated pattern. Then, the conjunctive, disjunctive supports and the

bond value of the retained candidates are computed. Thus, the uncorrelated candidates are also pruned. At the level n, the local minimal rare correlated patterns of size n are determined among the retained candidates. The local closed rare correlated patterns of size $n-1$ are also filtered. This process holds up when there is no more candidates to be generated.

3. The third and last step consists of filtering the global minimal rare correlated patterns and the global rare correlated patterns among the two sets of local minimal rare correlated patterns and local closed ones.

In what follows, we will explain more deeply these different steps of GMJP. The pseudo code is given by Procedure 1.

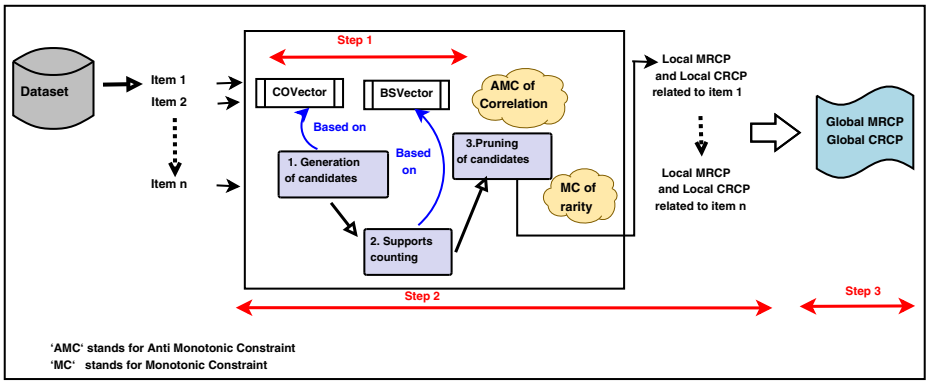


Fig. 3. Overview of GMJP when extracting the $\mathcal{RCPR}$ representation

First Step: The Power of the Bit Vectors and of Co-occurrent Vectors. Initially, the dataset is scanned in order to extract the items and to build, for each item, the bitset called here "BSVector". In fact, a bitset is a container that can store a huge number of bits while optimizing the memory consumption (For example, 32 elements are stored in a memory block of 4 bytes). Each block of memory is treated in just one CPU operation by a 32 bits processor. Therefore, we are very motivated for these kinds of structures within the GMJP algorithm in order to optimize the conjunctive and the disjunctive supports computations.

Then, the dataset is scanned again in order to identify, for each item I, the list of the co-occurrent items which corresponds to the items occurring in the same transactions as the item I. These latter ones are stored in a vector of integers, called here "COVector". We note that one of the main challenges of the GMJP algorithm is that it allows pushing two constraints of distinct types and to deliver the output with only two scans of the dataset. We uphold also that the bitsets, when incorporated into the mining process within the GMJP algorithm, sharply decrease the size of the memory required to store immediate results and significantly save execution costs.

Second Step: Getting the Local Minimal and the Local Closed Rare Correlated Patterns without Closure Computations. Worth of mention, the main thrust of the GMJP algorithm is to break the search space into independent sub-problems. In fact, for

Algorithm 1. GMJP

Data:
1. A dataset $\mathcal{D}$.
2. A minimal correlation threshold *minbond*.
3. A minimal conjunctive support threshold *minsupp*.
4. A specification of the desired result '$\mathcal{RCPR}$'.

Results: The concise exact representation $\mathcal{RCPR} = \mathcal{MRCP} \cup \mathcal{CRCP}$.

Begin

1. Scan the dataset $\mathcal{D}$ twice to build the BSVector and the COVector for all the items
2. For each item I
   (a) n =2;
   (b) Generate the candidates of size n using the COVector of I
   (c) **While** (*The number of the generated candidates is not null*) **Do**
      i. Prune these candidates w.r.t. the cross-support property of the *bond* measure
      ii. Prune these candidates w.r.t. the order ideal property of correlated patterns
      iii. Compute the conjunctive and disjunctive supports and the *bond* value of the maintained candidates
      iv. For each candidate C
         If (IsCorrelated(C) and IsRare(C)) **then**
         $/*$ Ckeck-Local Minimality of the candidate C $*/$
            – Update the set of Local Minimal Rare Correlated Patterns of size n
      v. Find Local Closed Rare Correlated Patterns of size $n-1$
      vi. n = $n+1$
      vii. Generate candidates of size n using the APRIORI-GEN procedure
3. Find all Global Minimal Rare Correlated Patterns
4. Find all Global Closed Rare Correlated Patterns
5. **Return** $\mathcal{RCPR}$;

End

each item I, a levelwise mining process is performed using the COVector containing the co-occurrent items of I. At each level n, starting by the second level, a set of candidates are generated, then pruned according to the different pruning strategies described previously. The minimal rare correlated patterns of size n associated to the item I are called **Local Minimal Rare Correlated Patterns** and they are determined by comparing their *bond* values to those of their respective immediate subsets. Similarly, the closed rare correlated patterns of size $n - 1$ associated to the item I are called **Local Closed Rare Correlated Patterns** and they are determined by comparing their *bond* values to those of their respective immediate supersets.

It is also important to mention that the implementation of the different stages of this second step (candidate generation, evaluation and pruning) was based on simple vectors of integers. Thus, we require no more complex data structure during the implementation of the GMJP algorithm. This feature makes GMJP a practical approach for handling both monotonic and anti-monotonic constraints even for large datasets.

One of the major challenges in the design of the GMJP algorithm is how to perform subset and superset checking to efficiently identify Local Minimal and Local Closed patterns? The answer is to construct and manage a multimap hash structure, [5] in order to store at each level n the rare correlated patterns of size n. This technique is very powerful since it makes the subset and the superset checking practical even on dense datasets.

Thus, our proposed efficient solution (as we prove it experimentally later) is to integrate both the monotonic constraint of rarity and the anti-monotonic constraint of correlation into the mining process and to identify the local closed rare correlated patterns without closure computing.

Third Step: Filtering the Global Minimal and the Global Closed Rare Correlated Patterns. After identifying the local minimal and the local closed rare correlated patterns associated to each item I of the dataset $\mathcal{D}$, the third step consists in filtering the $\mathcal{MRCP}$ set of Global Minimal Rare Correlated patterns and the $\mathcal{CRCP}$ set of Global Closed Rare Correlated patterns. This task is performed using two distinct multimap hash structures. In fact, for each local minimal rare correlated pattern LM previously identified, we check whether it has a direct subset (belonging to the whole set of local minimal patterns) with the same *bond* value. If it is not the case, then the local minimal pattern LM is a global minimal rare pattern and it is added to the $\mathcal{MRCP}$ set. Similarly, for each local closed rare correlated pattern LC previously identified, we check whether it has a direct superset (belonging to the whole set of local closed patterns) with the same *bond* value. If it is not the case, then the local closed pattern LC is a global closed rare pattern and it is added to the $\mathcal{CRCP}$ set of Closed rare correlated patterns.

In what follows, we illustrate with a running example of the GMJP algorithm.

5.2 A Running Example

Let us consider the dataset $\mathcal{D}$ given by Table 1. First, the BSVectors and the COVectors associated to each item of this dataset are constructed, as we plot by Figure 4. These BSVectors are next used to compute the conjunctive and the disjunctive supports. We have, for example, the item A which belongs to the transactions $\{1, 3, 5\}$ and the item C which belongs to the transactions $\{1, 2, 3, 5\}$. We, then, have $Supp(\wedge AC) = 3$ and $Supp(\vee AC)) = 4$.

The local minimal and the local closed correlated rare patterns associated to each item I of the dataset $\mathcal{D}$, are extracted. A detailed example of the process of the item A is given by Figure 5. The finally obtained $\mathcal{RCPR}$ representation, for *minsupp* = 4 and for *minbond* = 0.20, is composed by the following global minimal and global closed

[5] We used in our implementation the C++ STL Standard Template Library multimap.

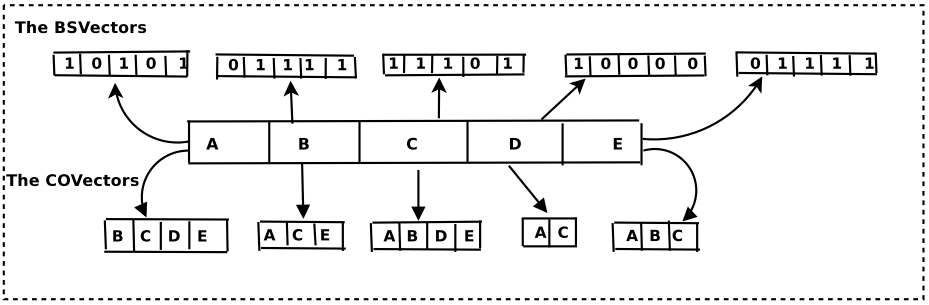


Fig. 4. The BSVectors and the COVectors associated to the items of the dataset $\mathcal{D}$

Jaccard patterns: $\mathcal{RCPR}$ = { $(A, 3, \frac{3}{3})$, $(D, 1, \frac{1}{1})$, $(AB, 2, \frac{2}{5})$, $(AC, 3, \frac{3}{4})$, $(AD, 1, \frac{1}{3})$, $(AE, 2, \frac{2}{5})$, $(BC, 3, \frac{3}{5})$, $(CD, 1, \frac{1}{4})$, $(CE, 3, \frac{3}{5})$, $(ACD, 1, \frac{1}{4})$, $(BCE, 3, \frac{3}{5})$ and $(ABCE, 2, \frac{2}{5})$}.

Last, it is important to notice that GMJP is not an exclusive approach in the sense that it can be coupled with other efficient approaches to mine statistically significant patterns.

In the next section we report our experimental study of the proposed GMJP algorithm.

6 Experimental Evaluation

Datasets and Experimental Environment: Experiments were carried out on different dense and sparse benchmark datasets [6]. All the tests were carried out on a PC equipped with a 2.40 GHz Intel Core TM $i3$ processor and 2.92 GB of main memory, running the Linux Ubuntu 10.04. Running times were averaged over 5 executions.

Protocol: Our objective is to prove, through extensive carried out experiments, the efficiency of the proposed GMJP algorithm while running the four different scenarios. Our first batch of experiments aims to build a quantitative comparison between the $\mathcal{FCP}$, the $\mathcal{RCP}$ sets and their associated condensed representations. Our second batch of experiments focus on studying running times.

Results: As sketched by Table 2, the concise representations $\mathcal{FCCPR}$ and $\mathcal{RCPR}$ present very encouraging reduction rates over several datasets and for different ranges of *minsupp* and *minbond* theresholds. We note that, the 'gain' corresponds to the reduction rate and is equal to : 1 - $\frac{|\mathcal{RCPR}|}{|\mathcal{RCP}|}$ for rare *Jaccard* patterns, and equal to 1 - $\frac{|\mathcal{FCCPR}|}{|\mathcal{FCP}|}$ for frequent ones.

[6] Available at *http://fimi.cs.helsinki.fi/data* and at http://archive.ics.uci.edu/ml.

We conclude, according to the results gived by Table 3 [7] , that the execution time varies depending on the number of distinct items of the considered dataset. This is explained by the principle of GMJP which is based on the idea of processing each item separately and based on the list of the co-occurrent of each item. For example, the computational costs are relatively high for the T40I10D100K dataset, and they are lower for the MUSHROOM dataset. This is explained by the fact that, the MUSHROOM dataset contains only *119* items while the T40I10D100K dataset contains *942* items. We note also that the highest execution times are obtained with the RETAIL dataset, since this latter contains a high number of distinct items, equal to 16,470.

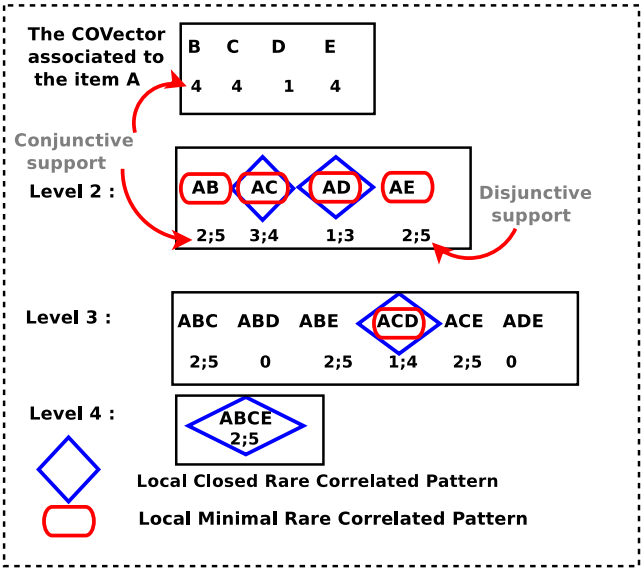


Fig. 5. Mining Local Minimal and Local Closed Rare Correlated Patterns for the item A

It is worth of mention that the computational time of the fourth scenario dedicated to the extraction of the $\mathcal{RCPR}$ are the highest ones. This can be explained by the fact that the extraction of the $\mathcal{RCPR}$ representation is an NP-hard problem since the localization of the associated two borders is a complex task. We also highlight that the performance results of the designed GMJP algorithm can not be compared to any approach of the literature. Indeed, the proposed approach is the first one dedicated to the extraction of *Jaccard* patterns in a generic way.

In the next section, we study the process of classification based on correlated association rules derived from the previous presented condensed representations.

[7] We note that 'S1' stands for the First Scenario, 'S2' stands for the Second Scenario, 'S3' stands for the Third Scenario and 'S4' stands for the Fourth Scenario.

Table 2. Effectiveness of *Jaccard* patterns mining on UCI benchmarks

Dataset	*minsupp*	*minbond*	# $\mathcal{FCP}$	# $\mathcal{FCCPR}$	Gain of $\mathcal{FCCPR}$	# $\mathcal{RCP}$	# $\mathcal{RCPR}$	Gain of $\mathcal{RCPR}$
MUSHROOM	30%	0.15	2, 701	427	**84.19**%	98, 566	1, 704	**98.27**%
	45%	0.15	307	83	**72.96**%	100, 960	1, 985	**98.03**%
PUMSB*	40%	0.45	10, 674	1646	**84.57**%	448, 318	3, 353	**99.25**%
	40%	0.50	9, 760	1325	**86.42**%	82, 413	3, 012	**96.34**%
CONNECT	10%	0.80	534, 026	15, 152	**97.16**%	56	56	**0**%
	50%	0.80	533, 991	15, 117	**97.16**%	91	91	**0**%
ACCIDENTS	40%	0.30	32, 529	32, 528	**0**%	117, 805	1, 722	**98.53**%
	60%	0.30	2, 057	2, 047	**0**%	148, 259	2, 743	**98.14**%

Table 3. Performance Analysis of GMJP on UCI benchmarks (time in second)

Dataset	Number of Items	Average *minsupp*	Average *minbond*	Average Time S1	Average Time S2	Average Time S3	Average Time S4
MUSHROOM	119	58%	0.30	7	11.4	20	19.6
		40%	0.57	3.75	5.25	11	709
ACCIDENTS	468	7.8%	0.50	709	703	793	784.2
RETAIL	16, 470	25.83%	0.50	5.83	13.16	1903	1902
T10I4D100K	870	5%	0.20	2	3	163	163
T40I10D100K	942	8.2%	0.50	148	182.6	491	490.4

7 Association Rules-Based Classification Process

We present in this section, the application of the $\mathcal{RCPR}$ and the $\mathcal{RFCCP}$ representations in the design of an association rules based classifier. In fact, we used the $\mathcal{MRCP}$ and the $\mathcal{CRCP}$ sets, composing the $\mathcal{RCPR}$ representation, within the generation of the generic [8] rare correlated rules. The $\mathcal{RFCCP}$ representation is used to generate generic frequent correlated rules, of the form $Min \Rightarrow Closed \setminus Min$, with Min is a minimal generator and $Closed$ is a closed pattern. Hence, we implemented a C++ program allowing the extraction of the correlated frequent minimal generators. Then, from the generated set of the generic rules, only the classification rules will be retained, *i.e.*, those having the label of the class in its conclusion part. After that, a dedicated classifier we designed is fed with these rules and has to perform the classification process and returns the accuracy rate for each class.

We report in Table 4 [9] the impact of integrating the correlation constraint for a fixed *minsupp* and *minconf* thresholds. We remark, for the frequent patterns, that while

[8] By "generic", it is meant that these rules are with minimal premises and maximal conclusions, w.r.t. set-inclusion.

[9] We note that Accuracy Rate = $\frac{NbrCcTr}{TotalNbrTr}$, with $NbrCcTr$ stands for the number of the correctly classified transactions and $TotalNbrTr$ is equal to the whole number of the classified transactions, and *minconf* corresponds to the minimum threshold of the confidence measure [1].

Table 4. Evaluation of the classification accuracy *versus minbond* variation for frequent and rare *Jaccard* patterns

Dataset	*minsupp*	*minconf*	*minbond*	# Exact Rules	# Approximate Rules	# Classification Rules	Accuracy rate	Response Time (sec)	Property of Patterns
WINE	1%	0.60	0	387	5762	650	97.75%	1000	Frequent
			0.10	154	2739	340	95.50%	13.02	Frequent
			0.20	60	1121	125	94.38%	1.00	Frequent
			0.30	20	319	44	87.07%	0.01	Frequent
ZOO	50%	0.70	0.30	486	2930	235	89.10%	40	Rare
			0.40	149	436	45	89.10%	3	Rare
			0.50	38	88	11	83.16%	0.01	Rare
			0.60	12	31	6	73.26%	0.01	Rare

increasing the *minbond* threshold, the number of exact and approximate association rules decreases while maintaining always an important accuracy rate. Another benefit for *Jaccard* measure integration, is the improvement of the response time, it varies from *1000* to *0.01* seconds. Whereas, for the rare patterns, we highlight that the increase of the *minbond* threshold induces a reduction in the accuracy rate. This is explained by the decrease in the number of the obtained classification rules.

Table 5. Evaluation of the classification accuracy of frequent patterns *vs* rare patterns

Dataset	*minbond*	*minsupp*	*minconf*	# Exact Rules	# Approximate Rules	# Classification Rules	Accuracy rate	Property of *Jaccard* patterns
WINE	0.1	20%	0.60	7	274	25	76.40%	Frequent
			0.80	7	86	10	86.65%	Frequent
			0.90	7	30	4	84.83%	Frequent
	0.1	20%	0.60	91	1516	168	**95.50%**	**Rare**
			0.80	91	449	84	92.69%	Rare
			0.90	91	100	48	91.57%	Rare
IRIS	0.15	20%	0.60	3	22	7	**96.00%**	**Frequent**
			0.95	3	6	3	95.33%	Frequent
	0.15	20%	0.60	17	32	8	80.06%	Rare
			0.95	17	7	5	80.00%	Rare
	0.30	20%	0.60	3	22	7	**96.00%**	**Frequent**
			0.95	3	6	3	95.33%	Frequent
	0.30	20%	0.60	8	14	4	70.00%	Rare
			0.95	8	6	3	69.33%	Rare
TICTACTOE	0	10%	0.80	0	16	16	**69.40%**	Frequent
	0.05	10%	0.80	0	16	16	**69.40%**	Frequent
	0.07	10%	0.80	0	8	8	**63.25%**	Frequent
	0.1	10%	0.80	0	1	1	**60.22%**	Frequent
	0	10%	0.80	1, 033	697	192	**100.00%**	**Rare**
	0.05	10%	0.80	20	102	115	**100.00%**	**Rare**
	0.07	10%	0.80	8	66	69	**97.07%**	Rare
	0.1	10%	0.80	2	0	1	**65.34%**	Rare

We note according to the results sketched by Table 5, that for the datasets WINE and TICTACTOE , the highest values of the accuracy rate are achieved with the rare correlated rules. Whereas, for the IRIS dataset, the frequent correlated rules performed higher accuracy than rare ones. In this regard, we can conclude that for some datasets,

the frequent correlated patterns have better informativity than rare ones. Whereas, for other datasets, rare correlated patterns bring more rich knowledge. This confirms the beneficial contribution of our approach in inferring new knowledge from both frequent and rare *Jaccard* patterns.

8 Conclusion and Future Works

We proposed, in this paper, GMJP the first approach to mine *Jaccard* patterns in a generic way (i.e., with two types of constraints: anti-monotonic constraint of frequency and monotonic constraint of rarity). Our approach is based on the key notion of bit-sets codification that supports efficient *Jaccard* patterns computation thanks to an adequate condensed representation of patterns. Experiments realised on several datasets show the efficiency of GMJP according to both quantitative and qualitative aspects. An important direction for future work is to extend our approach to other correlation measures [10,18,20,22] through classifying them into classes of measures sharing the same properties.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB 1994), Santiago, Chile, pp. 487–499 (1994)
2. Barsky, M., Kim, S., Weninger, T., Han, J.: Mining flipping correlations from large datasets with taxonomies. In: Proceedings of the 38th International Conference on Very Large Databases, VLDB 2012, Istanbul, Turkey, pp. 370–381 (2012)
3. Ben Younes, N., Hamrouni, T., Ben Yahia, S.: Bridging conjunctive and disjunctive search spaces for mining a new concise and exact representation of correlated patterns. In: Pfahringer, B., Holmes, G., Hoffmann, A. (eds.) DS 2010. LNCS, vol. 6332, pp. 189–204. Springer, Heidelberg (2010)
4. Bonchi, F., Lucchese, C.: On condensed representations of constrained frequent patterns. Knowledge and Information Systems 9(2), 180–201 (2006)
5. Booker, Q.E.: Improving identity resolution in criminal justice data: An application of NORA and SUDA. Journal of Information Assurance and Security 4, 403–411 (2009)
6. Bouasker, S., Hamrouni, T., Ben Yahia, S.: New exact concise representation of rare correlated patterns: Application to intrusion detection. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS, vol. 7302, pp. 61–72. Springer, Heidelberg (2012)
7. Ganter, B., Wille, R.: Formal Concept Analysis. Springer (1999)
8. Grahne, G., Lakshmanan, L.V.S., Wang, X.: Efficient mining of constrained correlated sets. In: Proceedings of the 16th International Conference on Data Engineering (ICDE 2000), pp. 512–521. IEEE Computer Society Press, San Diego (2000)
9. Jaccard, P.: Étude comparative de la distribution orale dans une portion des Alpes et des Jura. Bulletin de la Société Vaudoise des Sciences Naturelles 37, 547–579 (1901)
10. Kim, S., Barsky, M., Han, J.: Efficient mining of top correlated patterns based on null-invariant measures. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 177–192. Springer, Heidelberg (2011)
11. Kim, W.-Y., Lee, Y.-K., Han, J.: CCMine: Efficient mining of confidence-closed correlated patterns. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 569–579. Springer, Heidelberg (2004)

12. Koh, Y.S., Rountree, N.: Rare Association Rule Mining and Knowledge Discovery: Technologies for Infrequent and Critical Event Detection. IGI Global Publisher (2010)
13. Le Bras, Y., Lenca, P., Lallich, S.: Mining classification rules without support: an anti-monotone property of jaccard measure. In: Elomaa, T., Hollmén, J., Mannila, H. (eds.) DS 2011. LNCS, vol. 6926, pp. 179–193. Springer, Heidelberg (2011)
14. Lee, Y.K., Kim, W.Y., Cai, Y.D., Han, J.: CoMine: efficient mining of correlated patterns. In: Proceedings of the 3rd International Conference on Data Mining (ICDM 2003), pp. 581–584. IEEE Computer Society Press, Melbourne (2003)
15. Mahmood, A.N., Hu, J., Tari, Z., Leckie, C.: Critical infrastructure protection: Resource efficient sampling to improve detection of less frequent patterns in network traffic. Journal of Network and Computer Applications 33(4), 491–502 (2010)
16. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 3(1), 241–258 (1997)
17. Manning, A.M., Haglin, D.J., Keane, J.A.: A recursive search algorithm for statistical disclosure assessment. Data Mining and Knowledge Discovery 16(2), 165–196 (2008)
18. Omiecinski, E.: Alternative interest measures for mining associations in databases. IEEE Transactions on Knowledge and Data Engineering 15(1), 57–69 (2003)
19. Romero, C., Romero, J.R., Luna, J.M., Ventura, S.: Mining rare association rules from e-learning data. In: Proceedings of the 3rd International Conference on Educational Data Mining (EDM 2010), Pittsburgh, PA, USA, pp. 171–180 (2010)
20. Segond, M., Borgelt, C.: Item set mining based on cover similarity. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 493–505. Springer, Heidelberg (2011)
21. Soulet, A., Raissi, C., Plantevit, M., Crémilleux, B.: Mining dominant patterns in the sky. In: Proceedings of the 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, Canada, pp. 655–664 (2011)
22. Surana, A., Kiran, R.U., Reddy, P.K.: Selecting a right interestingness measure for rare association rules. In: Proceedings of the 16th International Conference on Management of Data (COMAD 2010), Nagpur, India, pp. 115–124 (2010)
23. Szathmary, L., Valtchev, P., Napoli, A.: Generating rare association rules using the minimal rare itemsets family. International Journal of Software and Informatics 4(3), 219–238 (2010)
24. Tanimoto, T.T.: An elementary mathematical theory of classification and prediction. Technical Report, I.B.M. Corporation Report (1958)
25. Tsang, S., Koh, Y.S., Dobbie, G.: RP-tree: Rare pattern tree mining. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 277–288. Springer, Heidelberg (2011)
26. Wu, T., Chen, Y., Han, J.: Re-examination of interestingness measures in pattern mining: a unified framework. Data Mining and Knowledge Discovery 21, 371–397 (2010)

Comparison Queries for Uncertain Graphs

Denis Dimitrov, Lisa Singh, and Janet Mann

Georgetown University
Washington, DC, USA 20057

Abstract. Extending graph models to incorporate uncertainty is important for many applications, including disease transmission networks, where edges may have a disease transmission probability associated with them, and social networks, where nodes may have an existence probability associated with them. Analysts need tools that support analysis and comparison of these uncertain graphs. To this end, we have developed a prototype SQL-like graph query language with emphasis on operators for uncertain graph comparison. In order to facilitate adding new operators and to enable developers to use existing operators as building blocks for more complex ones, we have implemented a query engine with an extensible system architecture. The utility of our query language and operators in analyzing uncertain graph data is illustrated using two real world data sets: a dolphin observation network and a citation network. Our approach serves as an example for developing simple query languages that enables users to write their own ad-hoc uncertain graph comparison queries without extensive programming knowledge.

Keywords: graph query language, comparison queries, uncertain graphs.

1 Introduction

Traditional graph models can be extended to incorporate uncertainty about vertex / edge existence and attribute values. Application data that could benefit from uncertain graph models include disease transmission networks with disease transmission probabilities, observed social networks with node existence probabilities, and physical computer networks with associated reliability probabilities.

A variety of tools, databases, algorithms, and research consider analysis of probabilistic data [16,26,27]. Analysis of graph data is a subject of another large, but mostly unrelated research area [8, 9, 11–13]. We are interested in combining both in a general way to give users the capability to examine and compare uncertain graphs.

While uncertain graph analysis and comparison is relevant to a number of different domains, we focus on two settings in this work, uncertainty occurring during scientific observation and uncertainty resulting from data analysis.

Observational Scientific Data: Observational scientists study animal societies in their natural settings, often, with the purpose of understanding the social relationships and behaviors within the society. Such social network data can be captured as a graph, where nodes represent observed animals and edges

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 124–140, 2013.

between nodes represent sightings of both animals together. A researcher observing a particular animal may be uncertain about its identification, features, or behavior. This uncertainty can be expressed as existence probabilities between 0 and 1, associated with nodes and/or edges, and categorical uncertain attributes, representing discrete probability distributions over the set of possible categorical attribute values. Analyzing and comparing uncertain graphs can be useful for answering questions about changes in animal sociality over time, similarities of local animal communities to the general animal population, differences among animal subgroups across locations, and observation bias across researchers.

Analysis Output Data: A second setting we consider involves machine learning algorithms generating uncertain graphs that can be used as the basis for prediction, generalization, and statistical analysis. One specific example is a *node labeling* algorithm. These algorithms can be used to predict the topics of each publication in a citation network, predict which customers will recommend products to their friends using a customer network, or predict the political affiliations of people in a social network [23]. They take as input a partially observed graph to predict the label (attribute value) of nodes in the graph. The output of such algorithms is an uncertain graph containing a probability distribution across the possible set of labels for each node. Comparing and contrasting these uncertain graphs to each other or to a ground-truth graph allows researchers to analyze the performance of different machine learning algorithms, experiment with a single algorithm under different assumptions, and examine the graph dataset by highlighting parts of data where the algorithms disagree in their predictions or perform poorly.

Contributions: Our focus is on uncertain graph analysis with an emphasis on comparison of them. The contributions of this work are as follows:

1. A basic SQL-like language, which incorporates uncertain graph analysis and comparison operators, while taking advantage of most of the SQL capabilities. The semantics of this proposed language is a combination between relational database and uncertain graph semantics, part of which is novel and part of which is necessary for the language to be applicable. Section 4 introduces the language.
2. A set of composable, comparative operators for uncertain graphs, where the previous literature focuses on single graph operators, on specific graph algorithms in the presence of uncertainty, or on operators for multiple certain graphs. We introduce operators for estimating similarity between graphs, nodes, edges, and their attributes, including finding a common subgraph that exists across two graphs containing edges with high certainty and identifying a set of nodes that have the same predicted node label across two uncertain graphs. Section 5 describes our proposed operators in more detail.
3. A novel system framework that (1) uses a combination of a layered and a service oriented architecture, and (2) is extensible, modular, and expandable, allowing for easy integration of new operations. The novelty of our design is the focus on extensibility and modularity. Traditionally, databases construct query trees whose set of possible operations is predefined. This

design allows for query optimization by applying a set of rewrite rules. Our approach provides a flexible mapping of operators to their implementation. The query engine can, therefore, support easy integration of new operators without affecting the existing ones and without requiring significant changes to the framework itself. Section 6 presents our high level system architecture emphasizing details related to operator extensibility and composition.

4. An implementation of our query framework and an empirical analysis using two real world data sets (presented in Section 7).

2 Related Literature

Storage, analysis, and manipulation of graph data is a vast area of interest for both the research community and the industry. In particular, there are multiple graph databases in existence: [2, 4–7], to name a few, offering efficient data storage and access, as well as scalability and transaction management. The query options sometimes include proprietary APIs (Neo4j Traverser) or proprietary SQL-like query languages [6]; in other cases ([1, 2, 5–7]) there is support for Gremlin [3] or SPARQL [20]. SPARQL is a query language for the RDF format with similarities to our approach in terms of semantics and SQL-like syntax, including joins and the capability to retrieve and combine data from several graphs. In comparison to our language, SPARQL offers more flexible pattern matching, but is more restricted in that its standard data types are XML-based, its set of operators is more limited, and it does not offer extended SQL-like constructs such as MERGE BY and SPLIT BY. Gremlin is a relatively simple but powerful language for graph traversal and manipulation supporting built-in functions such as union, difference, intersection that are applicable to graph comparison. Neither of these languages focus on uncertain graphs. Similarly, many of the languages suggested in existing research [8, 9, 11–13] often do not consider uncertainty during graph analysis and comparison across multiple graphs.

Querying similar graphs in graph databases has been studied in recent years [30]; however, existing works mainly focus on structural information and connectivity. Uncertainty is often incorporated in the context of specific algorithms [14, 15, 18, 19, 24, 32]. These problems are important in answering some of the possible uncertain graph queries, yet our goal is to create a more comprehensive set of albeit simpler uncertain comparison operators. Other researchers study uncertainty arising from approximate queries rather than uncertain data [29].

Probabilistic databases, on the other hand, typically support queries based on the concept of Possible World Semantics [16, 26, 27]. Recently researchers have extended the Possible World Semantics to uncertain graphs [31], [28]. While applicable for many problems, this concept is different from our focus on graph comparison regardless of the nature of the underlying probabilities. We do, however, build upon operators for comparison of probabilistic attributes [26], as they are applicable for uncertain graph attributes.

Finally, there are a number of visual graph tools, including [23], [10]. Excellent for visual comparison of uncertain graphs, they could complement rather than substitute the capability to execute user-defined queries.

3 Probabilistic Formulation

Throughout the subsequent sections we use the following background definitions, underlying assumptions, and notation. The central object of interest are uncertain graphs that represent a generalization of exact graphs, incorporating uncertainty about vertex / edge existence and attribute values. The existence probability of an edge is assumed to be conditional upon the existence of its endpoints. Uncertain attributes contain probabilities associated with each possible value from their domain, expressing the likelihood that the attribute takes on this particular value. To broaden the application of our model, we make no assumptions about the nature of probability values assigned to the graphs that we need to compare. In other words, the analyst can decide if the probabilities are marginal or posterior.

Uncertain Graph. An uncertain graph $G = (V, E, A^V, A^E, PA^V, PA^E)$ has a non-empty finite set of vertices, $V = \{v_1, \dots v_m\}$, and a finite set of undirected edges, $E = \{e_1, \dots e_n\}$, where each edge e_y is a pair of vertices, $e_y \in V \times V$, and $V \times V = \{(v_i, v_j) | v_i \in V, v_j \in V\}$; $A^V = \{A_1, \dots A_p\}$ is a set of (certain) attributes for vertices; $A^E = \{A_1, \dots A_q\}$ is a set of (certain) attributes for edges; $PA^V = \{PA_1, PA_2, \dots PA_r\}$ is a set of uncertain attributes for vertices; and $PA^E = \{PA_1, PA_2, \dots PA_t\}$ is a set of uncertain attributes for edges. The attributes are consistent across vertices and across edges respectively, i.e. all vertices have the same schema and so do all edges. We refer to both edges and vertices as *graph elements*.

Certain Attributes. The set of all certain attributes is defined as $A = \{A_1, A_2, \dots A_s\} = A^V \cup A^E$. Given an attribute $A_j \in A^V$, its domain D_j, and a vertex $v_i \in V$, we associate a value $p_k \in \{D_j\}$ with the pair (v_i, A_j) and denote it using the notation $a(v_i, A_j) = p_k$. As shorthand, we use a_{ij}. Similarly, $a(e_i, A_j) = p_k$. To express structural uncertainty, we designate one of the attributes in A^V and A^E to store our confidence about existence of the corresponding graph element: $\exists A_j : a_{ij} \in [0, 1], \forall i \in [1, m]$ and $\exists A_j : a_{ij} \in [0, 1], \forall i \in [1, n]$. Henceforth, we refer to this attribute as 'conf' or 'confidence'.

Uncertain Attributes. By analogy, the set of all uncertain attributes is $PA = \{PA_1, PA_2, \dots PA_o\} = PA^V \cup PA^E$. Uncertain attributes allow the data model to express semantic uncertainty in the graph. The value of an uncertain attribute PA_j is a set of pairs of each possible attribute value and a probability associated with each possible value. For example, an uncertain attribute *sex* with value domain $\{male, female\}$ reflects the researcher's uncertainty about the sex of the observed animal. For a specific vertex, the set of its value pairs could be $\{(male, 0.8), (female, 0.2)\}$.

More precisely, value domain VD_j is the constrained (discrete) domain of possible values associated with attribute PA_j. The value domain is ordered and we use the notation a_j^t to designate the t-th member of VD_j, where $t \in [1, |VD_j|]$.

Given an uncertain attribute $PA_j \in PA^V$ and a vertex $v_i \in V$, we associate a value $p_k \in \{PD_j\}$ with the pair (v_i, PA_j) and denote it using the notation $pa(v_i, PA_j) = p_k$. As shorthand, we use pa_{ij}. Similarly, given an attribute $PA_j \in$

PA^E and an edge $e_i \in E$, we associate a value $p_k \in \{PD_j\}$ with the pair (e_i, PA_j) and denote it using the notation $pa(e_i, PA_j) = p_k$. As shorthand, we again use pa_{ij}.

By analogy to using a_j^t to refer to members of value domain VD_j, the shorthand p_{ij}^t refers to the corresponding probability $f(a_j^t)$, associated with value a_j^t for vertex v_i. We define the set of uncertain attributes for a particular vertex v_i as $PA(v_i) = \{PA_j : PA_j \in PA^V \text{ and } pa(v_i, PA_j) \neq null\}$. Similarly, the set of uncertain attributes for a particular edge e_i is defined as $PA(e_i) = \{PA_j : PA_j \in PA^E \text{ and } pa(e_i, PA_j) \neq null\}$.

Assumptions. In comparing two graphs g1 and g2, we use the simplifying assumption that the following partial mapping exists between their elements: (1) the vertex mapping consists of a bijective mapping function for those vertices that are mapped, plus a set of unmapped vertices in each of the graphs $g1$ and $g2$; and (2) edge mapping is equivalent to vertex mapping, with the added constraint that edges $g1.e$ and $g2.f$ can be mapped to each other only if both of their endpoint vertices are also mapped. We refer to two graphs with this property as aligned graphs. Different alignment schemas may be possible. Unless stated otherwise, alignment is assumed to be based on element id: elements from graph $g1$ are mapped to elements with the same unique id in graph $g2$; they are unmapped if there is no corresponding element with the same id.

4 Query Language

Our approach was to take advantage of the SQL semantics by handling graphs, graph elements, and attributes using relations. We chose to base our query language on SQL because it is a mature, proven, and well-known language. While we do not claim that it is the best language for the purpose, we believe it is sufficient for expressing a wide range of uncertain graph comparison queries using our set of operators. Using these operators directly with traditional SQL was certainly another option; however, there are several disadvantages. On a syntactic and semantic level, a dedicated query language allows the flexibility for any modifications that best suit the specifics of uncertain graph analysis. On an implementation level, extending an existing SQL query engine effectively would mean using a relational database as storage for uncertain graph data. Instead, our goal was to develop an approach that borrows important SQL semantics without being restricted to a particular storage.

Our query language supports the major SQL operations, such as *SELECT*, *FROM*, *JOIN*, *WHERE*, *GROUP BY*, *HAVING*, and *ORDER BY*, introducing modifications and extensions to accommodate the specifics of graph comparison. For example, the *FROM* operation can extract individual nodes, edges, both nodes and edges, attributes from the specified graph (creating a tuple for each of them), or return the graph as a whole in a table as a single tuple. An example that returns a table containing edges from graph $g1$ with high existence probability along with the confidence of existence sorted by confidence is as follows:

```
SELECT e, conf(e)
FROM g1 TYPE edge AS e
WHERE conf(e) > 0.5
ORDER BY conf(e) DESC
```

We introduce two new operations, *MERGE BY* and *SPLIT BY*, to support collections that are returned by some common operators, such as $adjacentVertices()$. The *SPLIT BY c* operation is used to "flatten out" a relation, when column c may contain a set of values rather than a single value. In the returned relation, each tuple t in the source relation is replaced by a number of tuples, one for every value in the set of values from t in column c. *MERGE BY* is similar to *GROUP BY*, but retains the remaining columns, which are not part of the clause, "visible" to subsequent non-aggregate operators by collapsing the values from the original tuples into a collection.

While this approach has many strengths, the main advantages are the use of a familiar query language in SQL and the additional operation extensions for collections that improve the flexibility of querying uncertain graphs, without being restricted to a particular storage.

5 Proposed Operators

While we take advantage of the SQL-like operations to retrieve, filter, sort, group, and join data, individual operators are used within each of these clauses to specify the required behavior. As shown in the query example, the same operator can be re-used with several operations, subject to rules between aggregate vs. non-aggregate operators and operations. Because limited space precludes us from describing all operators within the text, we provide a list of operators, categorized based on their target, i.e. those that apply to attributes (Table 2), graph elements (Table 1), and ego-networks and graphs (Table 3). Each of these tables contains the name of the operator, a brief description, and in some cases, an example result based on the sample graph in Figure 1.

More specifically, because the attribute value represents a discrete probability distribution, the proposed *attribute operators'* functionality ranges from simply extracting the most / least probable value to analyzing the shape of the distribution. For example, $peakToNextDist()$ can be used to identify uncertain attributes with a dominant (peak) probability that significantly exceeds the probabilities for the remaining values. *Graph element operators* may be incorporated into a query such as the introductory example in section 4 to identify weak connections within a single graph or to compare the confidence of the corresponding elements across two aligned graphs, isolating the elements not only based on their low or high confidence, but also on whether the two graphs agree or differ significantly. Common usage examples of *graph* and *ego-net operators* are provided in Table 3.

We focus the remainder of this section on our similarity operators since they are most relevant to the uncertain graph comparison tasks outlined in the motivating examples.

Table 1. Graph Element Operators: Most of these operators serve to query and analyze the confidence of existence of a single graph element or relative to another vertex or edge. Other operators in this group aggregate the results from attribute-level operators for the given graph element.

Operator	Description
$conf()$	confidence of element's existence - $conf(v1) = 0.8$
$bin()$	true or false bin, corresponding to high or low $conf()$ relative to a threshold. Ex. $bin(v1, 0.5) = \text{true}$
$compBin()$	"high", "opposite", or "low", depending on the relationship between the output of the bin() operator applied to each of the two operands. Ex. $compBin(v1, v2) = high$
$magnitudeDiff()$	difference between confidence of existence of 2 elements. Ex. $magnitudeDiff(v1, v2) = 0.2$
$diffSignificance()$	whether the absolute value of magnitude difference is above a threshold. Ex. $diffSignificance(v1, v2, 0.1) = true$
$valueCertaintyScore()$	average $maxValueCertainty()$ of all uncertain attributes of the element. Ex. $valueCertaintyScore(v1) = 0.6$
$sim()$	similarity score between 2 elements of the same type (vertices or edges), typically in the range [0, 1]

Uncertain Attribute Similarity. The $sim()$ operator is one of our novel operators for comparing uncertain attributes pa_{ij} and pa_{lj}. In the proposed set of measures, similarity is classified as either structural or semantic. The former identifies the similarity between the general shapes of the two distributions, ignoring the attribute values and their arrangement relative to each other. For example, attribute $\{(a, 0.8), (b, 0.1), (c, 0.1)\}$ should be considered structurally equivalent to attribute $\{(a, 0.1), (b, 0.1), (c, 0.8)\}$, as both have a dominant value (peak) of 0.8. We support two structural similarity measures, entropy ratio and absolute distance ratio, where the entropy ratio compares the distribution spread for the specified uncertain attribute and the absolute distance ratio compares the magnitude of the distance between the different uncertain attribute values. For example, the absolute distance is calculated as: $AD(pa_{ij}) = \sum_{t=2}^{|VD_j|} |p_{ij}^t - p_{ij}^{t-1}|$ and by analogy, for pa_{lj}. To correctly reflect structural similarity through absolute distance, probability sets in both attributes must first be sorted.

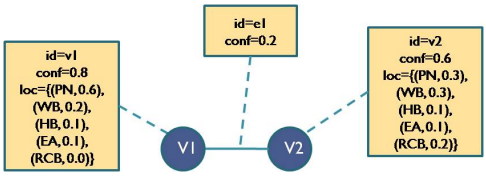


Fig. 1. Sample network for operator examples

Table 2. Uncertain Attribute Operators: These operators answer queries about values and probabilities associated with one or more uncertain attributes

Operator	Description
$mpv()$, $lpv()$	most / least probable attribute value. Ex. $mpv(v1.loc) =$ "PN"
$valueCertainty()$	probability that the attribute has a specific value from the domain. Ex. $valueCertainty(v1.loc, \text{"}WB\text{"}) = 0.2$
$maxValueCertainty()$, $minValueCertainty()$, $avgValueCertainty()$, $medianValueCertainty()$	max, min, mean, or median probability among all probabilities associated with an attribute. Ex. $maxValueCertainty(v1.loc) = 0.6$
$peakToAvgDist()$	difference between the max and the average certainty. Ex. $peakToAvgDist(v1.loc) = 0.4$
$peakToNextDist()$	difference between the max and the second-highest attribute value probability. Ex. $peakToNextDist(v1.loc) = 0.4$
$valueCertaintyDev()$	standard deviation of probabilities for an uncertain attribute
$valueCertaintyRange()$	difference between highest and lowest probability of an attribute. Ex. $valueCertaintyRange(v1.loc) = 0.6$
$sim()$	similarity score between two uncertain attributes of the same type, typically in the range [0, 1]. It is generally measured between the two sets of their respective attribute values and probabilities. The specific similarity measures are described in this section.

The semantic similarity, on the other hand, compares probabilities between the corresponding attribute values. An instance of an uncertain attribute can be represented as a histogram. We refer to each possible attribute value as a 'bin' in the histogram, conceptually containing the associated probability. This representation allows us to use a number of measures that have been proposed for histogram similarity. They generally fall into two categories [21]. The *bin-by-bin* similarity compares the contents of only corresponding bins, or in our case, probabilities for the same attribute values in two attribute instances. *Cross-bin* measures, on the other hand, compare non-corresponding bins. This is possible only if the *ground distance* between pairs of non-corresponding attribute values is known. In this work, we focus on the following bin-by-bin similarity measures [21]: (1) Default: $sim(pa_{ij}, pa_{lj}) = 1 - \frac{\sum_{t=1}^{|VD_j|} |p_{ij}^t - p_{lj}^t|}{2}$; (2) Minkowski-Form Distance; (3) Histogram intersection; (4) K-L divergence.

Ego Network Similarity. The $egoSim()$ operator uses a variety of similarity measures and algorithms depending on user-specified constraints and on ego network containment within the same or different graphs. For measuring similarity between two ego networks (or ego-nets), the two center nodes are mapped to each other, each of the non-center nodes from the first subgraph is mapped to 0 or 1 non-center nodes from the second subgraph, and vice versa. Depending on

Table 3. Graph and ego-net operators: Operators allowing for structural graph comparison based on graph alignment include: $intersect(g1, g2)$, $union(g1, g2)$, $difference(g1, g2)$, and $bidirectionalDifference(g1, g2)$. For example, by intersecting the ego networks (subgraphs consisting of the starting node v_i (center), all of its adjacent nodes, and all edges between them) of two specific dolphins in the same graph, the analyst can discover their common friends. The graph reconstruction operator, $toGraph()$, can be used in a query to derive a subgraph based on specified conditions. For example, to obtain a subgraph of high-confidence elements, it can be combined with the $bin()$ operator and MERGE BY clause. While this operator is not as sophisticated as pattern matching [13], it does provide the possibility of subgraph filtering based on a flexible set of conditions.

Operator	Description
$egoNet()$	given a vertex v_i, returns the set of vertices and edges that are part of v_i's ego-network, including v_i itself
$egoSim(v_1, v_2)$	similarity score between two ego-networks defined by their center vertices v_1 and v_2, respectively, typically in the range [0, 1]
$intersect()$, $union()$, $difference()$, $bidirectionalDifference()$	creates a new graph that represents, respectively, an intersection, union, difference, and bi-directional difference of two graphs
$toGraph()$	recreates a graph from a set of vertices and edges
$toElements()$	breaks down a given graph into a set of vertices and edges

existence and on alignment between the two ego-nets, similarity can be aligned and unaligned. In the aligned case, the mapping is determined by the alignment scheme. If no alignment scheme is chosen (not aligned case), the ego-nets' elements are mapped in a way that maximizes similarity.

Ego-net similarity can be structural, semantic or both. Structural similarity only takes into account the existence or confidence of existence of vertices and edges in each mapped pair between the two ego-nets, while ignoring attributes and their values. Semantic similarity, on the other hand, ignores confidence and derives the similarity score by only using similarity measures between the individual nodes and edges in the mapped pairs. Structural-semantic similarity is a combination of both.

We now intuitively describe different types of ego-net similarity. They are the cornerstone of our uncertain comparative operators, allowing researchers to better compare graph substructures, not just the entire graph or single graph elements. Due to space limitations, we cannot describe them more formally. We have an extended version of this paper that contains those details.

We propose the following three different structural similarity operators for uncertain graphs. *Topological similarity* compares the structure of the two ego-nets based on the existence of their elements, but not on confidence values associated with existence. *Probabilistic-topological similarity* takes into account the confidence values associated with edges and non-center vertices. *Comparison count*

is simply a count of aligned non-center nodes between the two ego-networks. It is useful when the researcher is interested in an absolute similarity measure, related to the size of the ego-networks, rather than in a ratio between 0 and 1 that is returned by the topological and probabilistic-topological similarity.

Semantic similarity takes into account uncertain attribute values and probabilities, rather than confidence of existence of mapped vertices and edges. In the aligned case, similarity is measured by aggregating similarities between pairs of attributes with the same name and definition, belonging to each pair of aligned vertices. In the unaligned case, alignment is chosen in an attempt to maximize the total similarity of all attribute pairs between aligned vertices - usually by greedy heuristics.

Depending on the number of attributes under consideration, the measure can be either single- or multiple-attribute. In both of those cases, similarity between a pair of uncertain attributes can be estimated using different measures. We propose two of them: mpv and distribution similarity. The former calculates the similarity between a pair of attributes to be 1 if there is either a full or a partial match between their sets of mpv values, and 0 otherwise. The latter represents the user's choice of one of the uncertain attribute similarity measures, described in a previous section. In the aligned case, attribute similarity is calculated as average pairwise similarity between mapped vertices.

In the unaligned case, we restrict the similarity measure to a single attribute for considerations of computational complexity. Even in the case of a single attribute, the brute force approach for finding the alignment that would maximize similarity is highly inefficient in some cases. In those cases, we propose using a greedy heuristics, similar to the merge-sort algorithm, that reduces running time but does not guarantee optimality. As in the aligned case, the user has a choice of two similarity measures, MPV and distribution based similarity.

Other Operators: In addition to operators related to uncertain graph comparison, the proposed query language supports general operators, most of which are commonly present in many other languages, including SQL, e.g. aggregate operators, logical operators, set operators, etc.

Route Operators: While path operators are central to graph query languages, their use is not as central as similarity for uncertain graph comparison. Some operators that are useful in this context include: comparing high confidence path existence between two nodes or ego-nets, comparing high confidence shortest paths, and comparing connected components when taking into account the confidence of existence of graph elements.

6 System Architecture

6.1 System Overview

Our highest priority design goals in developing the query engine architecture and prototype implementation include:

Extensibility. Because we intend to continue building upon our initial query language, allowing for extensibility at all levels was our highest priority. At

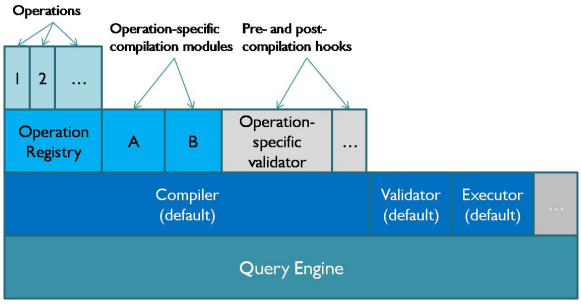


Fig. 2. Layered architecture

the lowest level, it must allow easy integration of any additional operators and operations. When concepts that do not fit in the existing implementation are introduced, for example, aggregate operators, it is desirable to minimize the required changes to the query processing framework. We refer to this as mid-level extensibility. At a high level, the design must provide room for new system capabilities, such as plugging in different data storage implementations.

Operator Composition. Operators sometimes re-use the functionality of other existing operators. For example, vertex and ego-network similarity operators build upon different attribute similarity operators. Because we anticipate that being a common situation, the system should provide re-use of existing operators to the programmer, who creates new operators. The user can also compose operators implicitly within the limits of the query language by creating expressions or within the limits of the pre-programmed sub-operator selections, such as choosing an underlying attribute similarity measure.

Adaptability. Capabilities to introduce future optimizations specific to our data model and query language without restricting the implementation to a particular platform or data storage.

To meet these goals, we use a combination of layered and service-oriented architecture, illustrated in Figure 2. The main component of this architecture is the query engine, a lightweight and generic platform for deployment of modules responsible for the individual steps in the query processing workflow, such as parsing, compilation, optimization, validation, and execution. The set of included modules, represented as services, is not pre-defined, making it different from traditional database query engines.

The engine offers two important capabilities: service configuration and service lookup. The former allows parameter tuning without code recompilation, including deploying the same implementation under different configurations. The latter allows flexible and dynamic linking of services, e.g. transparent replacement of the underlying data storage implementation. The individual modules are designed with the goal of decoupling them from each other and, in turn, they can be customized by plugging in implementations of their sub-components. For example, an operation registry is the sub-component that provides the default

mapping between operators and their compiled representations. For integrating simple operators, it is sufficient to add a reference to the registry.

The query execution process is as follows. It begins when the Parser module transforms the textual representation of a query into an abstract syntax tree (AST), which we refer to as *logical query*. The Compiler module translates the AST using a post-order traversal of operations in the logical tree into an internal representation suitable for optimization and execution. Next, the Optimizer module generates and evaluates several alternative execution plans, choosing the best one. The Executor is the key module where the operations and operators that make up the query are executed. The Validator module can be invoked at different stages to ensure compliance with the pre-defined rules. The Facade and Connector modules provide the interface for interaction between external systems and the query processing workflow. Data Store serves to retrieve the data requested in the query. Due to space limitations, we cannot go through the details of each step. Instead, we focus on how one develops and composes new operators given the extensible system design.

6.2 Developing and Composing Operators

Traditionally, databases construct query trees whose set of possible operations is predefined. This design allows for query optimization by applying a set of rewrite rules. Our approach differs because we provides a flexible mapping of operators to their implementation. This allows the query engine to support easy integration of new operators without affecting the existing ones and without requiring significant changes to the framework itself.

Developing an operator involves implementing a simple interface with two methods. The first method allows the Executor to set the operator's input parameters. Then, the second method is called, in which the operator performs its calculations over these supplied parameters and returns the result. In the simple case, no other code is required. Adding the operator to the configuration of the OperationRegistry is sufficient to incorporate it into the query language, as the registry is used for both compilation and execution. Composing an operator using operators that are already in the language is also straightforward, as the framework supports their lookup and execution from the dependent operator.

7 Empirical Evaluation

To show the utility and composition ability of our operators, we have integrated our query engine with Invenio [25], a visual analytic tool for graph mining. We use the two motivating scenarios presented in section 1 to highlight a subset of our operators.

7.1 Dolphin Observation Network

The Shark Bay Research Project studies dolphins in Shark Bay, Australia for over 30 years [17]. Our data set includes demographic data about approximately

800 dolphins, represented as graph nodes with certain attributes (id, conf, dolphin name, birth_date) and uncertain attributes (sex_code, location, mortality_status_code). Survey data about social interactions between these dolphins are captured as approximately 29,000 edges with attributes (id, conf).

Our team met with researchers on this project and developed a list of typical queries that observational scientists would like the capability to issue when analyzing these dolphin social network and its inherent uncertainty:

- Selecting the number of associates and sex composition of associates for male and female dolphins, respectively, using the most probable value of the *sex_code* attribute.
- Visualizing the union, intersection, difference, and bi-directional difference between the ego-networks of a particular dolphin during two different years, where the confidence of relationship existence is above a specified threshold.
- Finding the common associates (friends) of two specific dolphins with a relationship confidence above a certain threshold.
- Finding all dolphins having associates whose most probable location is different from their own.
- Calculating a measure of structural and semantic similarity between ego-networks of two particular dolphins.
- Selecting the subgraph that consists only of dolphins linked by observations with low confidence of existence (lower than a specified threshold). The results of this query tell researchers if observers are having difficulty identifying certain dolphins.

The query in Table 4 is an example that shows counts by sex of dolphins seen together (task 1). The inner query selects pairs of dolphins seen together and uses *SPLIT BY* to split into a set of rows the collection that is returned by the *adjacentVertices*() operator. The outer select produces counts for each possible sex combination, grouping the nodes based on the most probable sex_code. Researchers can use the resulting table to see that dolphins who are most probably males are seen together more often than any of the other combinations.

Table 4. Sample query and its result

```
SELECT sex, sexAdj, count(adj) AS cntFriends
FROM (
       SELECT *
       FROM
           (
              SELECT n, adjacentVertices(n) AS adj
              FROM g1 TYPE node AS n
           )
       SPLIT BY adj
       )
GROUP BY first(mpv(n.sex_code)) AS sex,
       first(mpv(adj.sex_code)) AS sexAdj
```

MALE	MALE	9930
MALE	FEMALE	6184
FEMALE	MALE	6184
FEMALE	FEMALE	6092

The second task focuses on determining the union, intersection, difference, and bi-directional difference between the ego-networks of a particular dolphin during two different years. It introduces a time component. The results for a particular dolphin are displayed in graph format in Figure 3. It is easy to see that the dolphin has almost as many new associates as repeat associates, i.e. occurring during both years. Researchers can then visually explore who these associates are, what sex they are, etc., to gain more insight about dolphin sociality.

To validate the significance of our similarity operators, we evaluate one of the more complex measures. We estimate the ego-network semantic similarity between dolphin *JOY* and other dolphins in the same graph, in absence of alignment, using the most probable location attribute. By picking dolphins with different characteristics, we can demonstrate the behavior and validity of the chosen similarity measure. For example, we discovered that the average ego-net similarity by location is twice as high for dolphins located in the same area as JOY, e.g. RCB: 0.28 vs 0.14. This is the expected result, since dolphins are likely to have associates mostly in their primary location.

To compare uncertain ego-nets, we randomly chose several dolphins from different locations with high and low similarity relative to *JOY's*. For every dolphin under consideration, we ran a query to retrieve their most probable location, their ego-network location similarity to that of *JOY*, and a breakdown by location of the dolphin's ego-network. The results are summarized in Table 5. They are consistent across the two cases of same and different location. Both *LITTLE* and *WHELK* differ from *JOY* in their most probable location; however, the ego-network's location composition between *LITTLE* and *JOY* results in a much higher similarity score. *PUCK* and *JOYSFRIEND* reside in the same location as *JOY*, share many associates with her, and have a very similar distribution of associates by location. These commonalities lead to a particularly high similarity score. *MYRTLE*, on the other hand, who only shares 88 out of 147 associates with *JOY* despite the same location, is average in similarity. For *WANDA*, the most probable location is a tie between WB and RCB, which is

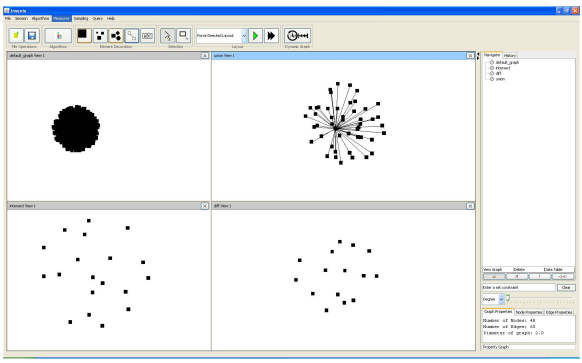


Fig. 3. Clockwise from upper left: complete dolphin network, union, intersection, difference of ego-networks of dolphin 'JOY' between years 2010 & 2009

also reflected in having associates from mostly those locations. This difference with *JOY's* ego-network again corresponds to the lower similarity.

Overall, examining different cases confirms that the similarity measure provides a relevant single numeric value that correlates with the semantic composition of a pair of ego-networks based on the chosen attribute. Researchers can use this simple result to identify and rank potentially similar ego-networks.

The query also shows operator re-use and composition from the user's perspective. By supplying context parameters, the user configures the general ego-net similarity operator. Specifying the mpv-based similarity measure and attribute name causes the similarity operator to re-use the $mpv()$ operator to retrieve the most probable attribute value.

Table 5. Ego-network similarity results

	JOY	LITTLE	WHELK	PUCK	JOYSFRIEND	MYRTLE	WANDA
RCB	211	125	1	171	172	92	56
EA	38	5	43	32	47	6	7
WB	28	48		7	8	45	33
HB		4				1	4
PN						5	
sim with JOY		0.56	0.14	0.75	0.78	0.44	0.32
primary loc	RCB	WB	EA	RCB	RCB	RCB	WB, RCB

7.2 Citation Network

In the second scenario, we examine the output of two different node labeling algorithms. For this analysis, we use the CiteSeer paper citation data set from [22]. It consists of 3312 scientific publications classified into one of six topics. In the citation network each publication is a node and each citation is an edge. We use partially observed citation data to predict the probability distribution of the topic attribute of each paper by applying two different classification algorithms. The queries of interest deal with understanding the similarities and differences between most probable node labels across the two classification algorithms and include:

- Selecting the papers, whose topic certainty is significantly higher in one uncertain graph when compared to the other.
- Selecting the papers, for which the predicted discrete probability distribution differs the most between the two graphs, using different attribute similarity measures, e.g. KL divergence, Minkowski-form distance, and histogram intersection.
- Counting the number of papers that are misclassified by both models.
- Selecting the papers, which are misclassified with high confidence by both classifiers.

Due to space limitations, we cannot show the actual queries in the paper. Some of the results we found using this data set are as follows: model 1 misclassified fewer documents (83) than model 2 (103); of the documents misclassified by both classifiers (65), both models misclassify them with the same label; and 8 of the 10 largest ego networks were in the area of information retrieval.

Because our experiments focus on expressibility, we do not include a performance evaluation. Yet it should be noted that all of the presented queries were completed within a couple of seconds on a personal computer with a moderate dual core 2.4 GhZ processor and 4 GB of memory. Optimizing performance is an important direction for future research.

These usage examples demonstrate how our language and implementation enable scientists to formulate a wide range of ad-hoc queries that can be used to analyze and compare uncertain graphs without the need for custom programming.

8 Conclusions

A need exists to compare graphs with uncertainty using a flexible set of operators that consider the graph structure, the graph semantics, and the inherent uncertainty in the application domain. In this paper, we develop a query engine with a SQL-type language, set of comparative operators for uncertain graphs, and an extensible system framework that combines layered and service-oriented architecture. Our language combines elements of relational, uncertain, and graph databases, with operators introduced especially for the purpose of uncertain graph analysis and comparison. We also present complimentary use cases and empirically illustrate their utility on two real world data sets.

There are many future directions, including optimizations on certain similarity measures, efficient measures for unaligned graphs, and measures that combine both similarity and routing.

Acknowledgements. The authors would like to thank the anonymous reviewers for their valuable feedback, Prof. Bala Kalyanasundaram and Prof. Richard Squier for helpful comments during the development of the system, and the Shark Bay Dolphin Research Project (SBDRP). This work was supported in part by NSF Grants #0941487 and #0937070 and ONR Grant #10230702.

References

1. Arangodb graph database, `http://www.arangodb.org/`
2. Dex graph database, `http://www.sparsity-technologies.com/dex`
3. Gremlin language for graph traversal and manipulation, `https://github.com/tinkerpop/gremlin/wiki`
4. Neo4j graph database, `http://neo4j.org/`
5. Oracle spatial and graph option, `http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/index.html`
6. Orientdb document-graph dbms, `http://www.orientechnologies.com/`
7. Titan graph database, `http://thinkaurelius.github.com/titan/`
8. Abiteboul, S., Quass, D., McHugh, J., Widom, J., Wiener, J.: The Lorel query language for semistructured data. International Journal on Digital Libraries 1, 68–88 (1997)
9. Angles, R., Gutierrez, C.: Survey of graph database models. ACM Computer Surveys 40, 1:1–1:39 (2008)

10. Cesario, N., Pang, A., Singh, L.: Visualizing node attribute uncertainty in graphs. In: SPIE VDA (2011)
11. Fortin, S.: The graph isomorphism problem. Technical report (1996)
12. Güting, R.H.: GraphDB: Modeling and querying graphs in databases. In: VLDB (1994)
13. He, H., Singh, A.K.: Graphs-at-a-time: query language and access methods for graph databases. In: ACM SIGMOD (2008)
14. Jin, R., Liu, L., Aggarwal, C.C.: Discovering highly reliable subgraphs in uncertain graphs. In: ACM SIGKDD (2011)
15. Jin, R., Liu, L., Ding, B., Wang, H.: Distance-constraint reachability computation in uncertain graphs. Proc. VLDB Endow. 4(9), 551–562 (2011)
16. Koch, C.: MayBMS: A system for managing large uncertain and probabilistic databases. In: Managing and Mining Uncertain Data. Springer (2009)
17. Mann, J., Team, S.B.R.: Shark bay dolphin project (2011), `http://www.monkeymiadolphins.org`
18. Papapetrou, O., Ioannou, E., Skoutas, D.: Efficient discovery of frequent subgraph patterns in uncertain graph databases. In: EDBT/ICDT (2011)
19. Potamias, M., Bonchi, F., Gionis, A., Kollios, G.: k-nearest neighbors in uncertain graphs. Proc. VLDB Endow. 3, 997–1008 (2010)
20. PrudHommeaux, E., Seaborne, A.: Sparql query language for rdf. W3C Recommendation 15 (2008)
21. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vision 40, 99–121 (2000)
22. Sen, P., Namata, G.M., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3), 93–106 (2008)
23. Sharara, H., Sopan, A., Namata, G., Getoor, L., Singh, L.: G-PARE: A visual analytic tool for comparative analysis of uncertain graphs. In: IEEE VAST (2011)
24. Shasha, D., Wang, J.T.L., Giugno, R.: Algorithmics and applications of tree and graph searching. In: ACM PODS (2002)
25. Singh, L., Beard, M., Getoor, L., Blake, M.B.: Visual mining of multi-modal social networks at different abstraction levels. In: Information Visualization (2007)
26. Singh, S., Mayfield, C., Mittal, S., Prabhakar, S., Hambrusch, S., Shah, R.: Orion 2.0: native support for uncertain data. In: ACM SIGMOD (2008)
27. Widom, J.: Trio: A system for data, uncertainty, and lineage. In: Managing and Mining Uncertain Data. Springer (2009)
28. Yuan, Y., Chen, L., Wang, G.: Efficiently answering probability threshold-based shortest path queries over uncertain graphs. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5981, pp. 155–170. Springer, Heidelberg (2010)
29. Zhou, H., Shaverdian, A.A., Jagadish, H.V., Michailidis, G.: Querying graphs with uncertain predicates. In: ACM Workshop on Mining and Learning with Graphs (2010)
30. Zhu, Y., Qin, L., Yu, J.X., Cheng, H.: Finding top-k similar graphs in graph databases. In: EDBT (2012)
31. Zou, Z., Gao, H., Li, J.: Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In: ACM KDD (2010)
32. Zou, Z., Li, J., Gao, H., Zhang, S.: Finding top-k maximal cliques in an uncertain graph. In: IEEE ICDE (2010)

Efficient Time Aggregation and Querying of Flashed Streams in Constrained Motes

Pedro Furtado

Univeristy of Coimbra,
Portugal
pnf@dei.uc.pt

Abstract. We propose and evaluate efficient, low-memory and low-consumption organization and query processing algorithms for a tiny Stream Management Engine (SME). The target sensor devices have low memory and computation capabilities, and high wireless data transmission costs. The SME represents data as streams, we discuss the approach and study how to optimize group-by aggregation over time-ordered data in that context, and to provide simple all-purpose group-by and join algorithms. We used an experimental testbed to evaluate the findings and prove the advantage of the alternatives and studies that we made.

1 Introduction

Low-cost autonomous wireless sensing devices can be deployed to collect sensor data, either logging it for later retrieval or sending it wirelessly, possibly in multi-hop fashion, to some computerized systems. Applications of such systems include environmental, medical, industry, smart buildings, warehouse tracking, transport logistics and surveillance. The devices themselves, a.k.a motes, have computational and wireless communication capabilities, being able to sense, store, read and route the data along one or more hops, and to interface with a collection device, typically through wireless or USB connections. Operation is frequently supported on top of a tiny operating system such as TinyOS [26] or Contiki [28]. Motes also have their own battery power source that provides autonomy and mobility. An example of a mote is the TelosB [25], with up to 48KB of code memory, which must fit the OS and applications, and up to 10KB or data memory.

Most mote designs come with support for an external flash memory with MB or GB of capacity, which enables logging and storage of large quantities of data. In that context, consider streams of sensed data being acquired and logged into the flash. We build a tiny Stream Management Engine (SME) that allows users to submit queries for the data. It does the usual relational algebra data manipulations of data management systems, such as selections, projections, aggregations, joins. The SME must fit into the tiny code base and be efficient. Questions that immediately pops-up are: whether a stream management system fits well into the memory and computational constraints of motes; whether it has acceptable performance? Are there advantages over pushing or pulling detailed log data to a collecting PC?

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 141–155, 2013.

We propose and evaluate compact and efficient algorithms, in particular time-ordered group-by aggregations (GBTime) over time-ordered storage, as well as more generic group-by and join algorithms over the tiny devices. In the process, we provide answers to the above questions.

Efficiency is measured according to three major metrics: code size, since it must fit constrained devices, query execution time, and energy consumed to execute the queries. Query execution efficiency guarantees that clients will have their responses quick enough, and consumed energy is paramount, since low consumption avoids frequent battery replacement, which is very undesirable in many practical deployments.

We propose and test the mechanisms for stream-based data management with the aggregation and joins over stream data, and we present the SME and query processing approach. The code size, efficiency and energy savings are evaluated experimentally, showing that, in spite of the very small data memory, it is advantageous to be able to store and query locally in the motes, reducing the amount of data that needs to be sent to PC. We also show that algorithms such as time-ordered aggregation have major performance and energy saving advantages.

This paper is structured as follows: section 2 reviews related work. Section 3 presents the SME model and shows how it is used to query over sensor networks. Sections 4 and 5 discuss query-processing algorithms, and Section 6 show experimental results. Section 7 concludes the paper.

2 Related Work

A mote is a node in a wireless sensor network that is capable of gathering sensory information, performing some computations, and communicating with other connected nodes in the network. For instance, TelosB features a IEEE 802.15.4/ZigBee compliant RF transceiver with integrated onboard antenna, achieving a 250 kbps data rate. It has a 8 MHz TI MSP430 microcontroller with 10kB of RAM and a 1MB external flash for data logging. It also has a programming and data collection USB interface, integrated sensors and interfaces for adding other sensors and actuators. TelosB runs tiny operating systems such as TinyOS [26] or Contiki [28] and is programmed using some C-based dialect that is compiled and loaded with the full code image. TelosB has a line-of-sight reach of about 100 meters. More generically, mote communication ranges go from anywhere between a few meters to kilometers, with a corresponding bill to pay concerning energy consumption. Power autonomy is an important aspect in Motes, since it gives them both breadth of deployment in any place and complete mobility. Communication is by far the costliest part in terms of power consumption. To decrease consumption, communication should be reduced. The radio is turned-on only long enough for the node to be able to participate in multi-hop network communication, if and when necessary. Motes typically have small data memories, but if the data is logged into flash and queries are posed against that flash memory, it is possible to reduce the amount of communication significantly. For instance, in TelosB [25], writing data to flash consumes 15 to 30 times less than

sending the same data to another node. This means that logging the data locally and aggregating it are important strategies for the SME.

Writing and reading data from flash is 10 to 100 times faster than exchanging and storing the data from the mote where it was acquired to a computer logging the data. In most cases, the data also needs to go through other nodes in a sensor network before it arrives at a sink node connected to the computer. If the data is logged locally and aggregated efficiently before it is sent over the air, considerable speedup can be achieved. Logging the data locally also results in further autonomy, with the possibility of using the motes as data loggers for longer spans of time. In certain cases, data indexes may reduce operation times significantly. While indexes designed for large databases assume block-based I/O, flash devices have specificities that modify the design, such as access characteristics and restricting modifications of written data. Indexes designed for flash memory include FlashDB [13], Lazy-Adaptive Tree [3], and MicroHash [19]. Because of NAND write restrictions, these approaches use log structures [26], which need large amounts of memory. We provide a simple and efficient approach for time-wise data and queries.

Sensor data management by means of middleware approaches is also related to this work. For clarity, we classify middleware as either intra-sensor network approaches, such as ours, and internet-based sensor data management, which pick sensor data from a sensor data source, then use internet-connected non-constrained computers with full sensor data management engines to integrate, compute and share the data. They do not instrument motes or work inside wireless sensor networks at all.

Intra-sensor network: In [1], the authors share a vision of storage-centric sensor networks where sensor nodes will be equipped with high-capacity and energy-efficient local flash storage, arguing that the data management infrastructure will need substantial redesign to fully exploit the presence of local storage and processing capability in order to reduce expensive communication. There are several works surveying middleware managing data over wireless sensor networks, such as [21, 22, 23]. Intra-network approaches include SQL-based solutions, such as TinyDB [12,8,18], Cougar [5] or PerLa[24]. These approaches provide a database front-end to a sensor network. For example, TinyDB runs a small database-like query engine at a sink node. All the remaining nodes in the WSN load the code that allows them to receive commands from the sink and reply with the data. These approaches do not provide a stream management engine for individual nodes, and do not focus on keeping the data in the nodes for longer.

Most other sensor network middleware approaches aim at simplifying programming and deployment, therefore they do not provide a local stream data management engine as ours does. The approaches typically allow users to express computations using a simplified model, or to load pieces of code (agents) for extending functionality. For instance, Kairos [10] offers a network-programming model that allows the programmer to express, in a centralized fashion, the desired global behavior of a distributed computation on the WSN. The Abstract Task Graph (ATaG) [4] is a data driven programming model for end-to-end application development on networked sensor systems. SINA (Sensor Information Networking Architecture) [14, 16] is a middleware architecture that abstracts the network of a sensor node as a distributed

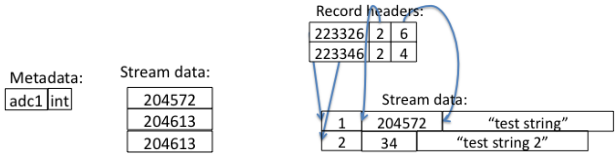
object for query operation, and task allocation. Data Service Middleware (DSWare) [11] takes a data-centric approach by defining the common data service and group based service parts of various applications. SensorWare [6] is a general middleware framework based on agent technology, where the mobile agent concept is exploited. Agents migrate to destination areas performing data aggregation reliably.

Internet-based: an example of an internet-based system is GSN [2], whose goal is efficient integration of multiple heterogeneous sensor data sources, with the capability of posing complex queries on the underlying data. GSN gets the streaming data from sensor sources using wrappers, and GSN's stream processing engine (which resides entirely outside of WSNs) is built on top of a relational database engine, storing and retrieving the streaming data during GSN's data processing. Other internet-based middleware architectures include Hourglass [15], HiFi [7] and IrisNet [9], which provide internet-based infrastructures for connecting sensor networks and sensor data sources. Semantic Streams [20] allows users to pose declarative queries over semantic interpretations of sensor data. Other popular internet-based sensor data management systems also include Pachube [29] and SensorCloud [30].

3 Overview of Stream Model and Its Use

A Stream Management Engine (SME) is installed in nodes and implements a query processor over ram and flash, and data exchanges between nodes. A stream has a metadata structure that is stored in flash or ram memory and defines attributes. Streams stored in memory are arrays of tuples, while flash-resident streams are stored in files (each stream is stored in one file). One typical use scenario is to log a stream of sensor data into flash and retrieve it later using queries. Another scenario is to acquire sensor data into a window in memory and to send the data to the PC when the window fills-up.

Physical storage on flash has two main possible organizations: constant- and variable-sized tuples. Figure 1 shows the metadata and data corresponding to both alternatives. Although SME can handle both, we focus on the simpler constant-sized tuples for our implementations on motes with small code and data memory.



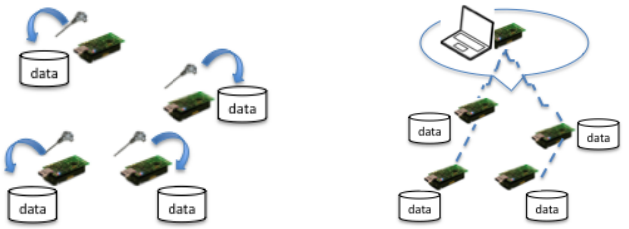
(a) Constant-sized (b) Variable-sized

Fig. 1. Metadata and Data: Constant and Variable-sized

If desired, it is possible to store one timestamp value per stream row. However, for streams representing periodic acquisitions (e.g. every minute) it is enough to store the pair (starting timestamp, acquisition rate). The timestamp of every tuples is determined from those. If the stream stops execution and restarts later, a new pair needs to

be added. This timestamp information of a stream is stored in a companion file that is called the timestamp index. In the case of variable-sized tuples, a b-tree index is used over the timestamp.

Queries are submitted using SQL dialect. Consider sensors deployed in a sensor data logging application, where nodes store collected information for some period of time. Since the amount of data may increase significantly over time, it needs to be logged into the flash. Figure 2 illustrates the logging (a) and querying (b) operations.



(a) Data acquisition and Logging (b) Querying and Retrieving data

Fig. 2. Logger Application Example

The next two commands set the timestamps of sensor nodes to a date and time and create the stream to log data. From then on, the sensors start logging temperature and light data into the stream that is resident on the flash.

```
Set timestamp of SensorNodes to ('10-SEP-2011-14:10:10','DD-MON-
YY-HH24:MI:SS');

Create stream environmentData in SensorNodes as
Select temperature, light From me
Sample every 1 minute
Window 1 year
Storage flash;
```

The following commands exemplify retrieving all the data for a specific month, and retrieving per-day temperature statistics (minimum, maximum and average).

```
Select nodeID, temperature, light, timestamp
from environmentData
Where time between todate(('01-DEC-2011-00:00:00','DD-MON-YY-
HH24:MI:SS') and todate('01-JAN-2012-00:00:00','DD-MON-YY-
HH24:MI:SS')

Select nodeID, min(temperature), max(temperature),
avg(temperature), todate(timestamp, ''DD-MON-YY')
from environmentData
group by todate(timestamp, ''DD-MON-YY');
```

A stream can be defined with a window of time-ordered tuples, and a query that is run against the window with a predefined period (the window size). A stream without a window is equivalent to a relational table with a set of time-ordered tuples in it. For instance, the window may collect 60 seconds of sensor data. Then, every 60 seconds,

it computes a per-10 seconds summary of the sensor data (average, variance and maximum), sends the computed summary to some destination, and empties the window for the next period of 60 seconds. The definition of the stream for this example is shown next:

```
Create stream pressure in SensorNodes as
Select avg(value),stdev(value),max(value)
From adc0
Sample every 1 second
Window 60 seconds
Group by 10 seconds;
```

This stream with the last 60 seconds of data fits nicely in the small memories of embedded devices. Every 60 seconds, its contents is sent to data consumers. A consumer stream is a stream that specifies this stream in the from clause, as shown next:

```
Create stream collectPressure in CollectionPC as
Select nodeID, *
From pressure;
```

The command syntax is summarized in the appendix.

4 Stream Relational Algebra and Algorithm

The sensor network is a distributed system with at least one SME in a sensor node and an SME with catalog and a Java console application in one PC. The catalog maintains all information on node configurations and status. Queries are submitted through the console. The query is pre-parsed into a query bytecode and nodes run the query and return the result to the caller. In the case of a stream with a window, when the window fills-up the query is ran and results forwarded to registered consumers (other streams).

The constrained SME version should occupy very small amounts of code and data memory, we will describe its query processing algorithms.

The base query-processing algorithm of Figure 3 works on a row-by-row fashion, retrieving one tuple at a time, applying selection and projection restrictions on the row and outputting the results if the row is not excluded by evaluated conditions. The select clause contains a set of expressions (e.g. stream attributes, parameters, constants, function calls such as todate(), aggregation functions applied to attributes, or simple expressions). These are pre-parsed in the console application into a bytecode that represents the select fields to be interpreted by the mote. Examples of node parameters that can be included in queries include “nodeID” or sensor identifiers. Where conditions are either “operand operator operand” expressions (binary) or “operand operator” expressions (unary). Operands are (simple) expressions, and operators are a set of possible operators (e.g. “>”,”<”,”=”,”!=”,”>=”,”<=”). Multiple where conditions can be “anded” or “ored”.

In the figure, the temporary aggregation structure A maintains additive quantities (sum s, square sum ss, maximum, minimum and number of tuples processed n) that allow aggregations to be computed after all the tuples were processed. For instance, the maximum and minimum are given directly from the current maximum and mini-

mum in the structure, the average is a sum divided by the number of tuples, and the variance is (ss-(s*s)/n)/n.

The query processing algorithm shown in the figure requires only a minimal amount of memory. It needs one tuple for input stream data, about 100 B for keeping metadata for each stream, few bytes for local variables used during query processing, space for the aggregation structure A (less than 50 B), and space for the output buffer O that holds result tuples. This buffer is flushed into network messages as soon as there are enough tuples to fill a packet payload, to be sent to the destination computer. This way, O needs only a packet payload size (about 100 B in telosB). We show results on the memory space that was consumed in the experimental section.

```
O= temporary tuple space for output tuples;
A=Aggregation structure, a temporary structure for computing aggrega-
tions;
1. Scan stream, tuple-by-tuple:
For each tuple,
     Apply selection operations (early-select) (where clause conditions)

     If selection operations evaluate to false (tuple will not contri-
     bute to output),
        go to step 2 with next tuple

     For each select clause field,
        If field is a constant, output it to a temporary output tuple
        space O;
        If the field is attribute, copy its value in current tuple to O;
        If the field is a function applied to an attribute, call the
        function with the attribute value of current tuple, output the
        result to O;
        If the field is an aggregation (e.g. sum, count, avg, max, min),
        the attribute value of the current tuple updates A, a temporary
        aggregation computation structure for that attribute (an aggre-
        gation hashmap);
        If (O already fills a network packet), fill the packet and send
        the results, emptying O)
2. End of query:
If the query is an aggregation, compose final output from aggregation
structure.
```

Fig. 3. Base Query Processing Algorithm

5 Constrained Group by and Join

The objective concerning constrained group by and join algorithms is to devise efficient solutions that may be run entirely in very small amounts of data memory, and the code should fit into the code memory of motes.

Sensor data is stored in stream format in monotonically increasing timestamp order, and it is very frequent to aggregate by time units. Therefore, we take advantage of the timestamp-order to define a simple Group-By Time approach with minimal memory requirements that is based on ordered aggregation. Only for the more generic case when the group-by attributes are not ordered we apply an external sort prior to using the same ordered aggregation algorithm. In the case of Join, a sort-merge join is implemented, with both relations participating in the join being sorted, followed by a merge-join. In this section we describe the algorithms, which are evaluated in the experimental section.

5.1 Time-Interval File Seek Index

Consider a query retrieving logged data for a time interval. Since the stream data is time-ordered and the stream has a timestamp index (section 3), the appropriate offset in the stream is calculated from the index and the query timestamp interval start. Tuples are then read until the timestamp interval end is reached. In the experiments we include a comparison of this with full table scan (fts) in the context of processing queries over datasets in TelosB motes.

5.2 GroupBy Time (GBTime) and GB-ALL

The objective of this algorithm together with the time-interval file seek is to run fast, save battery and to use the minimum possible amount of data memory, so that the algorithm can run efficiently in 2KB or less of data memory. The algorithm keeps a single aggregation computation structure (A) in data memory and is useful for grouping into time-intervals and to aggregate all the dataset. It is also used as the second step of the all-purpose group-by algorithm given in the next sub-section.

Consider an acquisition stream, that is, a stream that results from acquiring sensor data periodically, such as the example of section 3. The time granularity of the stream is given by the sensor-sampling rate (or the rate at which it receives data from another stream), and time-aggregated queries aggregate into some other time granularity. The Group-By Time algorithm of Figure 4 executes when a time argument is used in the group by clause. If the where clause contains a time interval condition (alone or "anded" with other conditions), the file seek index (section 5.1) is used to avoid full table scans. Then the algorithm simply scans the dataset tuples within the time interval specified in the where condition, while updating the aggregation computation for the current group. When the group changes (group time boundary is passed), the group aggregation is computed and it switches to compute the next group. The algorithm is described next.

```
   timeF: the time format string used in the query groups
                                ('DD-MON-YY,HH' in the above example);
   timeG: the current group identified as a string;
   aggregationStructure: A(timeG, s=0,ss=0,max=-1,min=MAXVAL,n=0);
                         (sum s, square sum ss, maximum max, minimum min,
                         and number of tuples processed n)

GBTime Algorithm:
0. timeG="";
1. If a time interval specified in the where clause restricts the interval that must be considered,
        Seek the position on flash corresponding to the start of the
        time interval specified in the where clause.
2. Scan the tuples one-by-one while the tuple timestamp is lower than the upper bound on time interval or the end of the dataset is met. Evaluate where conditions on the tuple, if tuple is excluded continue (2.) with next tuple;
        if todate(timestamp, timeF) for the tuple equals timeG
                update aggregation structure variables by adding the value(s) from the tuple;
```

```
      else // ended computing group aggregation for timeG
                compute select aggregations and expressions from A and
                output to O;
                if O fills packet, send packet and empty O;
                reset A structure for next group;
                timeG= new timeF;
3. Send O;
4. End.
```

Fig. 4. Group-By-Time Algorithm (GBTime)

GBTime is evaluated in the experimental section and its performance and energy consumption is compared with alternatives.

Resource use:

I/O (flash): n = number of tuples in dataset, nI in time interval

Constant-sized = nI (Variable-sized = log n + nI)

Minimum data memory:

sizeOf(A) + sizeOf(Tuple) + sizeOf(O), where O is the temporary output buffer that can be flushed whenever needed.

5.3 All-Purpose GroupBy (GB) and Join

An all-purpose Group-By is given for processing aggregations over generic non-stream-ordered attributes. A Sort-Group By algorithm is given. Similarly, an all-purpose Join is given with the Sort-Merge-Join algorithm. These will be slower when temporary flash-space is needed, but will handle generic aggregation and join operations. Figure 5 shows the Sort-GroupBy algorithm that was implemented. In Step 1 (external sort), the data set is sorted by the group-by attributes using an external sort (flash memory). Step 2 (re-)uses the GBTime algorithm of section 5.2, replacing time attribute values by the group-by attribute values in the algorithm. This way, the grouping of the sorted data can be done with small amounts of data memory, and re-utilizes the aggregation algorithm of GBTime (small code image).

We denote as GB-fts the GB algorithm doing a full-table scan and GB-idx a version using the timestamp index when one exists and a where clause restricts retrieval over a time interval.

```
GB-fts and GB-idx Algorithms:
Step 1. External Sort (simplified for the sake of brevity):
S= Sort buffer, should fit in memory, S=empty initially
For all tuples of dataset
   Apply where conditions, if tuple excluded by where conditions, con-
   tinue (1.) with next tuple;
   Project attributes and add remaining tuple values to S;
   If S full, apply in-mem sort algorithm, store as runfile and empty S;
For each input tuple from all runfiles
   Output next tuple in sort order and read next tuple from the runfile
   of the chosen tuple;
   Output tuples are flushed to flash when output buffer fills up, emp-
   tying the buffer;
```

Fig. 5. Sort-Group By Algorithm for SME

Resource use:
IO (flash): n = number of tuples in dataset, σ is where selectivity
n + 3σ(n) (read dataset, write runfiles, read runfiles, write sorted).
Minimum data memory: sizeOf(S)
For step 2, Sorted Group-By, see section 5.2, replacing time by group-by attribute values.

In the case of the Join algorithm shown in Figure 6, for the sort-merge join both datasets need to be sorted. The same external sort algorithm is used, then the algorithm reads sequentially tuples from both datasets simultaneously, outputting matches.

```
Sort-Merge-Join of datasets A and B
Run External Sort on A and on B to order by join attribute(s), resulting
in sortedA and sortedB (external sort Algorithm given above)
While there are input tuples from sortedA or sortedB
   If join attribute values for sortedA and sortedB match,
     Compose output tuple to O, from the sortedA and sortedB tuples;
     If O fills a packet, compose the packet and send it, then empty O.
     Retrieve next sortedA and sortedB tuples;
   Else
     Retrieve next tuple from either sortedA or sorted, by getting the
     smallest of the two based on the sort order. Replace the corres-
     ponding input tuple;
```

Fig. 6. Sort-Merge Join for SME

Resource use:
IO (flash): n = number of tuples in dataset, σ is where selectivity
IO(sort A) + IO(sort B) + σAnA+ σBnB
Minimum data memory: sizeOf(S) for sorts, then sizeOf(A Tuple) + sizeOf(B Tuple) + sizeOf(O).

6 Experiments

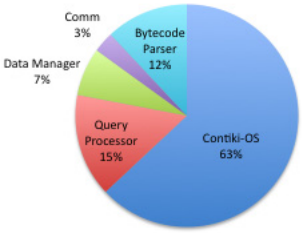
We have setup a set of experiments to test the approach and algorithms. Those were based on developing and running the SME for a TelosB mote with flash-stored streams, queried from a PC. Each TelosB has 48KB of ROM, 10 KB of data memory, 1024 MB of flash and 2 AA 800 mAh batteries. The PC is 2.53 GHz Intel Core 2 Duo, 4 GB 1067 MHz DD33 memory, running Windows 7, a Java Virtual Machine and a Java version of SME. The commands and data go through a TelosB sink node connecting to the gateway PC.

6.1 Code size and Data Memory

Figure 7 shows the size of the code composing the SME developed for TelosB. As shown in the figure, the total code of the SME occupies about 11KB, which together with about 20KB of the Contiki code size results in 31 KB for the total code size. This fits well into the 48KB of the code memory of this particular type of mote. Figure 7(b) shows that 63% of the code is occupied by the OS and that the second largest amount is the query processor (15%), followed closely by the query parser (12%).

Module	Code Size (B)
Contiki-OS	19710.81
Query Processor	4693.05
Data, Window and Sensor I	2190.09
Communications	938.61
Bytecode Parser	3754.44
TOTAL	31287

(a) Code Size of Modules (Bytes)



(b) code Size (%)

Fig. 7. Code Size of Modules

Figure 8 shows the minimum and maximum amount of memory occupied by the SME while it was running a query aggregating flash data using the algorithm GBTime. The temporary buffer (O) was set to 500 Bytes. We can see that the memory occupied was between 1.1 KB and 3 KB, fitting in the available data memory.

6.2 Time to Aggregate in Different Contexts

Figure 9 shows the time taken to aggregate in SME TelosB in memory (SME-ContikiTelosB) versus the time to aggregate when the dataset is on the flash (SME-TelosBflash) versus the time taken to aggregate the same amount in the computer Java version of the SME. Window sizes of 10, 50 and 100 were tried. The results show that memory operation was 5 to 6 times faster than flash on the TelosB and that aggregation on the PC was much faster.

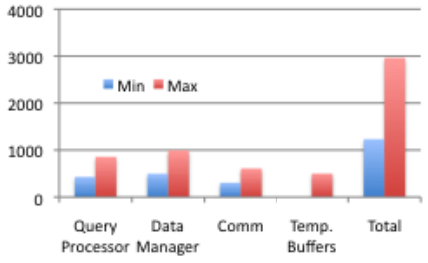


Fig. 8. Memory Occupancy SME (B)

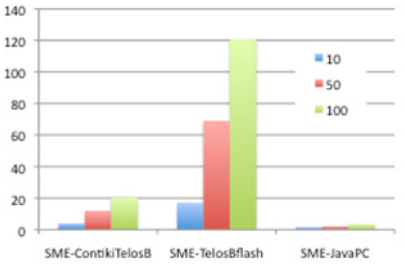


Fig. 9. Op. times (sec) vs windows size

6.3 GroupBy (GBTime, GB) over Time-Order

This experiment evaluates both efficiency and energy consumption of GBTime (section 5.2), GB-fts and GB-idx (section 5.3), and it also compares those with retrieving the data and sending it directly to be processed in the PC (Read&Send). We have loaded a dataset representing 6 months of per-minute sensor data. The "full table scan" alternatives (GB-fts and Read&Send(All)) ran over one-year dataset. Battery lifetime is measured in number of times the query can be submitted repeatedly before

the battery is depleted. To get it, energy consumption was measured for the query execution using Energest on Contiki, then the number of queries for battery depletion was calculated considering two typical 1.5V 800mA AA batteries and a cutoff voltage of 1.8 V.

Figure 11 shows the execution times of aggregation plus sending the results to the PC. The per-minute dataset was aggregated per-hour over a time interval of three months.

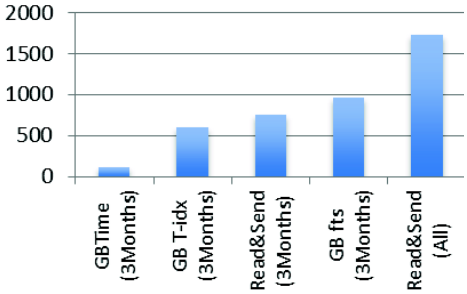


Fig. 10. Execution Times for Experiment 1 (secs)

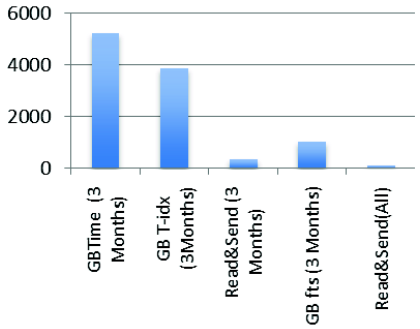


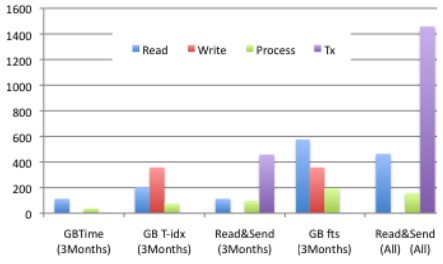
Fig. 11. Lifetime for Experiment 1

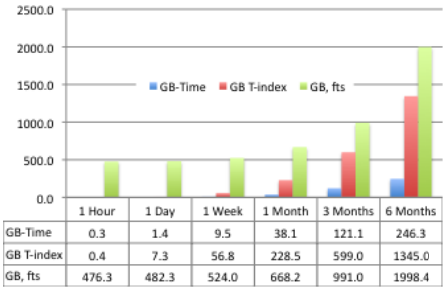
The GBTime algorithm took 121 seconds (about 2 minutes) to compute the three months results and send them to the collecting PC. This compares very favorably with GB-idx. It is also shown that if the data is simply retrieved to the PC without aggregation, it takes more than 12.5 minutes (748 secs), and full year data takes almost half an hour to collect. The time taken by GB-fts is worse than the time taken to Read&Send over 3 months of data because GB-fts is scanning the whole dataset (1 year), so GB-fts should be compared with Read&Send (All) that is also doing an fts.

As conclusion, GBTime is very efficient, and it is important to have a timestamp index to handle queries over time intervals faster.

Figure 12 shows the expected lifetime measured in number of runs of the query before batteries are depleted. The results prove that there is a great advantage in terms of lifetime in executing queries locally to summarize data before sending to the PC. The Read&Send queries resulted in orders of magnitude lower lifetimes than the aggregating queries. This is because data transmission consumes much more energy than computing or accessing flash memory.

Figure 13 is a breakdown of the time spent in each type of operation for each query type. Read&Send spends most time transmitting a large number of tuples to the collecting PC, and group-by algorithms (GB) other than GBTime spend significant time reading and writing to the flash, due to sort operations and, in the case of GB fts due to the full scan of the stream on flash. Figure 14 shows the time taken to execute an aggregation query using each alternative (GBTime, GB-idx, GB-fts) when the time interval (in the where clause) increases from 1 hour to 6 months. Once more the results show the advantage of GBTime and also show the relevance of the timestamp index (GB-idx vs GB-fts) in (time-)selective queries.





	1 Hour	1 Day	1 Week	1 Month	3 Months	6 Months
GB-Time	0.3	1.4	9.5	38.1	121.1	246.3
GB T-index	0.4	7.3	56.8	228.5	599.0	1345.0
GB, fts	476.3	482.3	524.0	668.2	991.0	1998.4

Fig. 12. Breakdown of Time in Operations (secs) **Fig. 13.** Varying Where Conditions

6.4 All-Purpose Group-By and Join

For evaluation of performance and battery lifetime of GB and Join algorithms (section 5.3), we ran experiments grouping and joining non-ordered datasets A and B with 5K, 10K, 25K and 50K rows. The group-by attribute in the datasets was chosen to generate a number of distinct groups that is 10% of the number of tuples. Figure 15 shows that Read&Send was faster than GB&Send for this test. This is because GB&Send has more overhead with sorting the dataset, computing aggregations and sending the results to the PC than Read&Send retrieving all tuples and sending them to the PC. Figure 16 shows lifetime for the same experiment. It is interesting to see that although GB&Send was slower than Read&Send, in terms of lifetime it lasts more than twice. This is because Read&Send transmits 90% more tuples, and transmission consumes much more power than computation or accessing the flash memory.

In what concerns the comparison with Join, it is interesting to compare the height of the upper limit of the dashed line on top of GB&Send with the execution time for Join. Join is processing two datasets with the same size (e.g. 5K with 5K), while the other alternatives are processing only one such dataset (e.g. 5K). The dashed lines are twice the execution time, which corresponds to processing (5K plus 5K). The execution time of GB&Send on the two datasets is very similar to the execution time of Join. Both cases sort both datasets, and both have to retrieve then every row of both datasets and do some processing with them. Finally, Join has to send the joined tuples, and GB has to send the aggregated tuples.

7 Conclusion

In this paper we proposed algorithms for a stream management engine to process data efficiently in constrained mote devices, including a very efficient time-ordered group-by processing solution, and all-purpose group by and join algorithms. We also proposed the Stream Management Engine approach to deal with the data. We have argued that the approach is very useful to allow applications to log data and to query the data efficiently, and that it is prepared to process streams in-memory and in flash and to send data between nodes. We evaluated the approach experimentally and against

alternatives, showing that the devised algorithms and Stream Management approach fits the memory of constrained devices and processes efficiently, while saving energy.

We have proposed and tested a base set of algorithms. In the future, we expect to test other forms of indexing and processing algorithms.

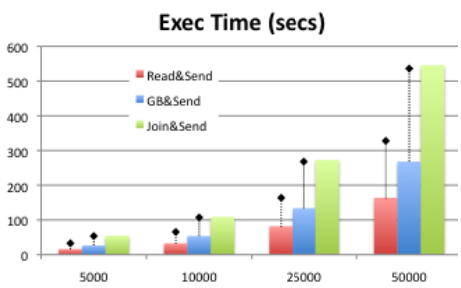


Fig. 14. Performance – GroupBy and Join

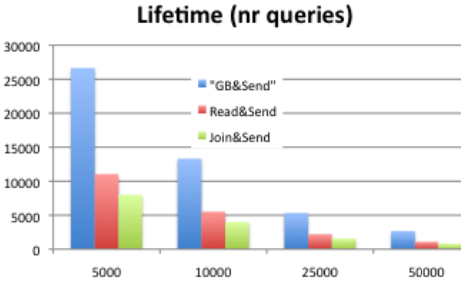


Fig. 15. Lifetime – GroupBy and Join

References

1. Diao, Y., Ganesan, D., Mathur, G., Shenoy, P.J.: Rethinking Data Management for Storage-centric Sensor Networks. In: CIDR, Asilomar, USA, pp. 22–31 (January 2007)
2. Aberer, K., Hauswirth, M., Salehi, A.: Infrastructure for data processing in large-scale interconnected sensor networks. In: Mobile Data Management, Germany (2007)
3. Agrawal, D., Ganesan, D., et al.: Lazy- adaptive tree: An optimized index structure for flash devices. In: Proceedings of IC Very Large Data Bases (VLDB), Lyon, France (August 2009)
4. Bakshi, A., et al.: The Abstract Task Graph: A Methodology for Architecture-Independent Programming of Networked Sensor Systems. In: Proc. EESR (2005)
5. Bonnet, P., Gehrke, J., Seshadri, P.: Towards sensor database systems. In: Proceedings of the Second International Conference on Mobile Data Management (2001)
6. Boulis, A., et al.: Design and implementation of a framework for efficient and programmable sensor networks. In: Proc. MobiSys (2003)
7. Franklin, M., Jeffery, S., Edakkunni, A., Hong, W., et al.: Design Considerations for High Fan-in Systems: The HiFi Approach. In: CIDR (2005)
8. Gehrke, J., Madden, S.: Query Processing in Sensor Networks. IEEE Pervasive Computing 3(1), 46–55 (2004)
9. Gibbons, P.B., Karp, B., Ke, Y., Nath, S., Seshan, S.: IrisNet: An Architecture for a World- Wide Sensor Web. IEEE Pervasive Computing 2(4) (2003)
10. Gummadi, R., Gnawali, O., Govindan, R.: Macro-programming wireless sensor networks using *kairos*. In: Prasanna, V.K., Iyengar, S.S., Spirakis, P.G., Welsh, M. (eds.) DCOSS 2005. LNCS, vol. 3560, pp. 126–140. Springer, Heidelberg (2005)
11. Li, S., et al.: Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. In: Proc. Int. Workshop on Information Processing in Sensor Networks (2003)
12. Madden, S., Franklin, M., et al.: TinyDB: an acquisitional query processing system for sensor networks. ACM Trans. on Database Systems 30(1), 122–173 (2005)

13. Nath, S., Kansal, A.: FlashDB: Dynamic self-tuning database for NAND flash. In: International Conf. on Information Processing in Sensor Networks Cambridge, USA (April 2007)
14. Shen, C.C., et al.: Sensor Information Networking Architecture and Applications. E Personal Communications Magazine 8(4), 52–59 (2001)
15. Shneidman, J., Pietzuch, P., et al.: Hourglass: An Infrastructure for Connecting Sensor Networks and Applications. Technical Report TR-21-04, Harvard University, EECS (2004)
16. Srisathapornphat, C., et al.: Sensor Information Networking Architecture. In: Proc. Int. Workshops on Parallel Processing (2000)
17. Rosenblum, Ousterhout, J.: The design and implementation of a log structured file system. In: ACM Sympo. on Operating Systems Principles, Pacific Grove, USA (1991)
18. Woo, A., Madden, S., Govindan, R.: Networking support for query processing in sensor networks. Commun. ACM 47(6), 47–52 (2004)
19. Zeinalipour-Yazti, Lin, S., et al.: MicroHash: An efficient index structure for flash-based sensor devices. In: USENIX FAST 2005, San Francisco, CA, USA (2005)
20. Whitehouse, K., Zhao, F., Liu, J.: Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data. Wireless Sensor Networks, 5–20 (2006)
21. Yoneki, E., Bacon, J.: A survey of Wireless Sensor Network technologies: research trends and middleware's role. Tech. R. of Univ of Cambridge, UCAM-CL-TR-646 (2005)
22. Wang, M.M., Cao, J.N., Li, J., et al.: Middleware for wireless sensor networks: A survey. Journal of Computer Science and Technology 23(3), 305–326 (2008)
23. Mottola, L.: Programming Wireless Sensor Networks: From Physical to Logical Neighborhoods. PhD Thesis, Politecnico di Milano, Italy (2008)
24. Schreiber, F.A., et al.: PERLA: a Data Language for Pervasive Systems. In: Sixth International Conf. on Pervasive Computing and Communications, Hong Kong, pp. 282–287 (2008)
25. Polastre, J., Szewczyk, R., Culler, D.E.: Telos: enabling ultra-low power wireless research. In: IPSN 2005. IEEE, Los Angeles (2005)
26. Levis, P., Madden, S., et al.: The Emergence of Networking Abstractions and Techniques in TinyOS. In: NSDI 2004, pp. 1–14. USENIX (2004)
27. `http://www.arduino.cc/`
28. Dunkels, A., Grönvall, B., Voigt, T.: Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In: LCN 2004 (2004) ISBN 0-7695-2260-2
29. Pachube [Pachube], `https://cosm.com/`
30. SensorCloud [SC], `http://www.sensorcloud.com/`

A Appendix – SME Query Expressions [nodeID| nodeSet] =NODES

```
[Create Stream Xpto [in NODES as]
Select [select expressions][in NODES ]
From [ sensorID | streamName | me ] [Where clause]
[Group by clause][sample clause] [window clause] [storageclause]

Update stream sensorID | stream [in NODES ]
set [set expressions] [Where clause];

Insert into stream sensorID | stream [in NODES] values [values];

Del stream sensorID|stream [in [nodeID|nodeSet]][Where clause];
```

Approximate OLAP Query Processing over Uncertain and Imprecise Multidimensional Data Streams

Alfredo Cuzzocrea

ICAR-CNR and University of Calabria,
87036 Cosenza, Italy
cuzzocrea@si.deis.unical.it

Abstract. A *novel framework for estimating OLAP queries over uncertain and imprecise multidimensional data streams* is introduced and experimentally assessed in this paper. We complete our theoretical contributions by means of an innovative approach for *providing theoretically-founded estimates to OLAP queries over uncertain and imprecise multidimensional data streams* that exploits the well-recognized *probabilistic estimators theory*. Finally, we provide an experimental assessment and analysis of the performance of our framework against several classes of synthetic data stream sets.

1 Introduction

Modern *data stream applications* [3] and systems are more and more characterized by the presence of *uncertainty* and *imprecision* that make the problem of dealing with *uncertain and imprecise data streams* a leading challenge in data stream research. Uncertainty and imprecision in data streams may occur due to several reasons, such as intrinsic properties of physical parameters, transmission errors, human errors, and so forth. This issue has recently attracted a great deal of attention from both the academic and industrial research community, as confirmed by several research efforts done in this context [7,18,2,8,19]. Uncertain and imprecise data streams arise in a plethora of actual application scenarios ranging from *environmental sensor networks* to *logistic networks* and *telecommunication systems*, and so forth.

While some recent papers have tackled the problem of efficiently *representing, querying and mining uncertain and imprecise data streams* [7,18,2,8,19], to the best of our knowledge, there not exist papers dealing with the problem of *efficiently OLAPing* [15] *uncertain and imprecise multidimensional data streams*, with explicit emphasis over *multidimensionality of data* (streams). In order to fulfill this relevant gap, in this paper we first introduce the problem of *estimating OLAP queries over uncertain and imprecise multidimensional data streams*, which can be reasonably considered as the first research attempt towards the definition of OLAP tools over uncertain and imprecise multidimensional data streams exposing *complete* OLAP functionalities, such as *on-the-fly data summarization* (e.g., [12,9]), *indexing* (e.g., [11]), and *OLAM* primitives.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 156–173, 2013.

With this goal in mind, in this paper we introduce the problem of answering OLAP queries over uncertain and imprecise multidimensional data streams, and propose a framework capable of efficiently and effectively providing *theoretically-founded estimates* to such class of queries. The solution to the problem of OLAPing uncertain and imprecise multidimensional data streams proposed in our research builds on some previous results that have been provided by recent research efforts. Particularly, [4], which focuses on static data, introduces a nice *Probability Distribution Function* (PDF) [20]-based model that allows us to capture the uncertainty of *OLAP measures*, whereas the imprecision of OLAP data with respect to *OLAP hierarchies* available in the multidimensional data stream model is meaningfully captured by means of the so-called *possible-worlds semantics* [4]. This semantics allows us to evaluate OLAP queries over uncertain and imprecise static data, while also ensuring some well-founded theoretical properties, namely *consistency*, *faithfulness* and *correlation-preservation* [4]. Similarly to the PDF-based model [4], the possible-worlds semantics approach is also exploited in our research, and specialized to the more challenging issue of dealing with uncertain and imprecise multidimensional data streams (contrary to the static case investigated in [4]).

2 A Probabilistic Data Model for Uncertain and Imprecise Multidimensional Data Streams

In this Section, we formally provide our proposed data model for uncertain and imprecise multidimensional data streams. This model relies-on and extends the research proposed in [4], which focuses on OLAP over uncertain and imprecise static data, like those one can find in *probabilistic relational database systems* [14], by introducing a nice data model and a theoretically-sound possible-worlds semantics. Particularly, [4] states that two specific occurrences of *data ambiguity* can arise in a typical OLAP scenario over static data: (*i*) uncertainty of (OLAP) measures, and (*ii*) imprecision of dimensional members of (OLAP) hierarchies.

We first specialize both these cases of OLAP data ambiguity in the particular context of dealing with multidimensional data streams by means of a meaningful running example. Then, formal definitions capturing these ambiguities are provided as well.

Consider a large retail company selling computer components in USA throughout several branches located in different USA states. First, note that, due to the networked structure of the company, sale data sent by different branches to the company head-office periodically can be reasonable assumed as streaming data. Also, these data are clearly multidimensional in nature, according to several attributes that *functionally* model typical *sale facts*. Some example attributes are the following: (*i*) *Store*, which models the store where the sale occurred; (*ii*) *Time*, which captures the time when the sale occurred, (*iii*) *Product*, which models the sold product, (*iv*) *Customer*, which captures the customer that purchased the product. These attributes play the role of *OLAP dimensions* for the multidimensional data stream model of the running example. In addition to this, OLAP hierarchies are associated to these dimensions. An OLAP hierarchy describes the hierarchical relationship among *dimensional members*

of the model [15]. Dimensional members are defined on top of functional attributes of the target OLAP analysis. For instance, consider the dimension *Store* of the running example. Here, a possible hierarchy associated to *Store* is: *State* → *City* → *Store*. A possible instance of this hierarchy is: (*i*) *California* → {*Los Angeles* → {*Computer Parts*, *PC Components*}, *San Francisco* → {*Computer Parts*, *PC Components*}}.

The main company is interested in performing on-the-fly OLAP analysis of sales across the different branches. As a consequence, attribute *Sale*, which records the sale of a product p to a customer c occurred in a certain store t during a certain day d, is the *OLAP measure* of interest for the target analysis. Due to data ambiguity, uncertainty of the measure *Sale* derives from the fact that data stream readings do not record the exact value of the sale (e.g., 50 $), but rather they record a confidence interval within which the possible value ranges (e.g., [45, 55] $) with a certain probability p_s, such that $0 \leq p_s \leq 1$. For the sake of simplicity, assume that the uncertainty of *Sale* is due to transmission errors over the network.

Due to data ambiguity, imprecision of OLAP dimensions derives from the fact that data stream readings can record *any* of the dimensional members of the available hierarchies at *any* of the hierarchical levels, instead than dimensional members of the *leaf level always* (as happens in the presence of *precise* OLAP data). Intuitively enough, data stream readings recording values at the leaf level of the whole OLAP hierarchy are defined as *precise readings* (i.e., the related OLAP level is *univocally determined*), whereas data stream readings recording values at any *intermediate level* of the whole OLAP hierarchy are defined as *imprecise readings* (i.e., the related OLAP level is *not* univocally determined). For instance, in our running example target data stream readings could record the dimensional member *California* (see the possible instance of the hierarchy associated to the dimension *Store* above) so that it is not possible to precisely and univocally determine the store where the related sale occurred effectively, as this store could both be *Computer Parts* or *PC Components*, alternatively. Again, for the sake of simplicity, assume that the imprecision of *Store* is due to transmission errors over the network.

Fig. 1 shows a two-dimensional OLAP view on multidimensional data streams generated by the different branches of the company. This view is exploited by the main company with the aim of making (on-the-fly) OLAP analysis of data streams via a set of OLAP queries posed against the view. Particularly, the target OLAP view is built by simultaneously combining the dimensions *Store* and *Product* available in the whole multidimensional model of the case study, enriched by the respective hierarchies. This view originates a continuous answer over multidimensional data streams (see Sect. 1). In Fig. 1, for the sake of simplicity, for each hierarchy defined on the dimensions *Store* and *Product*, respectively, dimensional members of the leaf level are represented by white circles, and the name of the corresponding dimensional attributes is not shown. Grey circles in Fig. 1 represent instead data stream readings that populate the view. Recall that data stream readings arrive at different rates and arrival times, so that our running example considers the case of a *snapshot* of the OLAP view over multidimensional data streams at a certain time t_V.

Due to imprecision of OLAP hierarchies, data stream readings can record values at any level of OLAP hierarchies associated to the dimensions of the view. For instance, reading $r_{s,i} = \langle 6789, [30, 46], 0.7, 177, S_i, M_i \rangle$ with identifier $ID_{s,i} = 6789$, which has

been originated at timestamp $t_{s,i}$ = 177, models a sale fact whose value ranges over the interval [30, 46] $ with probability $p_{s,i}$ = 0.7. This fact is related to the sale of the monitor M_i performed in the store S_i. Therefore, this is a precise data stream reading. Similarly, data stream reading $r_{s,j}$ = ⟨6795, [38, 52], 0.3, 188, S_j, M_j⟩ is precise as well. Consider instead data stream reading $r_{s,l}$ = ⟨6801, [75, 94], 0.9, 196, *CA*, *Laptop*⟩. At timestamp $t_{s,l}$ = 196, this reading records a sale fact whose value ranges over the interval [75, 94] $ with probability $p_{s,l}$ = 0.9. This fact is related to the sale of *a certain* product belonging to the (OLAP) group/class *Laptop* occurred in *a certain* store located in the (OLAP) group/zone *California*. Since *Laptop* and *California* are both intermediate dimensional members, then this is an imprecise data stream reading. As a consequence, $r_{s,l}$ is represented in Fig. 1 in terms of a dash square.

Inspired by [4], in our research we formally define an uncertain and imprecise *N*-dimensional data stream *s* as the following tuple:

$$s = \left\langle ID_s, [v_{s,min}, v_{s,max}], p_s, t_s, a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \right\rangle \tag{1}$$

such that: (*i*) ID_s is the (absolute) identifier of *s*; (*ii*) $v_{s,min}$ is the lower bound of the confidence interval associated to the possible value of *s*; (*iii*) $v_{s,max}$ is the upper bound of the confidence interval associated to the possible value of *s*, such that $v_{s,min} < v_{s,max}$; (*iv*) p_s is the probability with which the value of *s* ranges over the interval $[v_{s,min}, v_{s,max}]$; (*v*) t_s is the timestamp at which *s* is recorded; (*vi*) $a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}}$, with $k_j \in \{0, 1, \ldots, N-1\}$ and $n \leq N$, is a set of dimensional members associated to *s*, each one belonging to a certain OLAP hierarchy of the underlying *N*-dimensional data stream model.

A data stream reading $r_{s,j}$ of a data stream *s* is defined as the atomic element of a(n) (unbounded) *sequence* produced by *s*, $R_s = r_{s,0}, r_{s,1}, r_{s,2}, \ldots$, such that $j \rightarrow \infty$.

Looking at models, (1) can be considered as the *data stream model schema* of *s*, while each data stream reading $r_{s,j}$ in R_s can be considered as an *instance* of *s*, i.e. a possible realization of *s*, still adhering to the model (1).

Given a data stream *s* and a buffer *b* having size $B > 0$, $R_s^B = r_{s,j}, r_{s,j+1}, r_{s,j+2}, ..., r_{s,j+B-1}$ denotes a *B*-bounded sequence of data stream readings $r_{s,j}$ (of *s*) stored within *b*, such that $\left|R_s^B\right| < B$. As highlighted in Sect. 1, several of today data stream management systems make use of buffers for query optimization purposes (e.g., [1]). The information content of probabilistic measures of R_s^B can be nicely described by a *discrete interval-confidence-based* PDF P_s^B, defined as follows:

$$P_s^B[k] = \sum_{j=0}^{B-1} \left\langle [v_{s,j,min}, v_{s,j,max}], p_{s,j} \right\rangle \cdot \delta[k] \tag{2}$$

wherein: (*i*) *B* denotes the buffer size; (*ii*) $[v_{s,j,min}, v_{s,j,max}]$ denotes the confidence interval associated to the reading $r_{s,j}$ (of *s*); (*iii*) $p_{s,j}$ is the probability with which the value of $r_{s,j}$ ranges over the interval $[v_{s,j,min}, v_{s,j,max}]$; (*iv*) $\delta(\bullet)$ denotes the *Dirac impulse* [20].

To give an example, Fig. 2 shows a 5-bounded PDF associated to the OLAP measure *Sale* of the running example. Here, $P_s^B[2]$ models the fact that the measure

value ranges over the interval [25, 30] $ with probability $p_2 = 0.5$, whereas $P_s^B[4]$ models the fact that the measure value ranges over the interval [45, 50] $ with probability $p_4 = 0.1$.

3 A Data-Driven Approach for Computing the Probabilities of Imprecise Multidimensional Data Stream Readings

In Sect. 1, we have clearly highlighted that computing the probabilities of imprecise multidimensional data stream readings is the basic issue to be faced-off in order to achieve the definition of a possible-worlds semantics for OLAP over uncertain and imprecise multidimensional data streams. Moreover, it has been put in emphasis that the approach proposed in [4] is not feasible for the case of streaming data.

Inspired by these main motivations, in this research we propose an innovative approach to solve the relevant problem of achieving the definition of a suitable possible-worlds semantics for OLAP over uncertain and imprecise multidimensional data streams that is based on the well-known evidence stating that *OLAP data* (*static data as well as streaming data*) *are usually clustered and* (*highly*) *correlated in nature* [10,5,6,16]. On the basis of this evidence, the main idea of the approach we propose consists in *computing the probability of the confidence interval of an imprecise multidimensional data stream reading to be "close" to confidence intervals of its neighboring precise multidimensional data stream readings*, fixed the multidimensional range of the imprecise multidimensional data stream reading, at a certain OLAP hierarchical level, as *its reference neighborhood*. We name as *possible-worlds probabilities* these probabilities. In the following, we provide our approach for computing such probabilities.

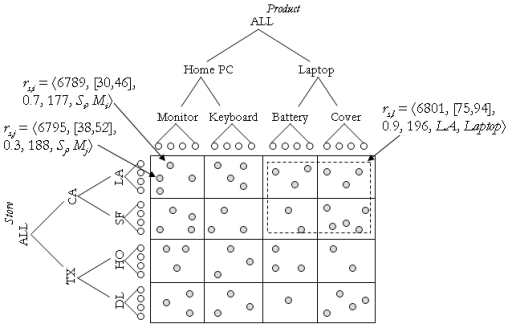


Fig. 1. The two-dimensional OLAP view over uncertain and imprecise multidimensional data streams of the running example

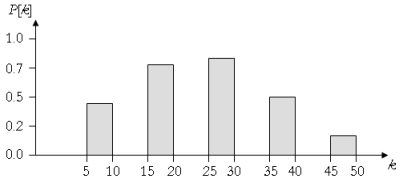


Fig. 2. A 5-bounded PDF for the OLAP measure *Sale* of the running example

Consider a multidimensional OLAP view V over the target multidimensional data streams. Let S be the schema of the view V defined as the tuple: $S = \langle m, d_0, d_1, \ldots, d_{N-1} \rangle$, such that m is the measure of interest and $D = \{d_0, d_1, \ldots, d_{N-1}\}$ the set of dimensions. Let $H = \{h_0, h_1, \ldots, h_{N-1}\}$ be the set of hierarchies of S, such that h_i is the

hierarchy associated to the dimension d_i. Let $Leaf(h_i)$ denote the leaf level of dimensional members of the hierarchy h_i. Also, let P_M denote the depth of the whole OLAP hierarchy of V.

Consider an imprecise multidimensional data stream reading populating V $r_{s,l} = \langle ID_{s,l}, [v_{s,l,min}, v_{s,l,max}], p_{s,l}, t_{s,l}, a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$, such that: (*i*) $k_j \in \{0, 1, \ldots, N\text{-}1\}$; (*ii*) $n \leq N$; (*iii*) for each a_{s,l,k_j} in $r_{s,l}$, $a_{s,l,k_j} \notin Leaf(h_{k_j})$. We denote as $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ the multidimensional range associated to $r_{s,l}$, and as $\langle d_0, d_1, \ldots, d_{N-1} \rangle$ the *base multidimensional range* of the view V, which is obtained by combining *all* the dimensions of V at the leaf levels of their respective hierarchies. Let P_L denote the depth of the OLAP level of $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ and P_M the depth of the OLAP level of $\langle d_0, d_1, \ldots, d_{N-1} \rangle$, respectively (it should be noted that P_M is also the depth of the whole OLAP hierarchy of V). Since $a_{s,l,k_j} \notin Leaf(h_{k_j})$ for each a_{s,l,k_j} in $r_{s,l}$, $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ is *contained* by $\langle d_0, d_1, \ldots, d_{N-1} \rangle$, or, alternatively, $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ is a *proper sub-set* of $\langle d_0, d_1, \ldots, d_{N-1} \rangle$, i.e. $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle \subset \langle d_0, d_1, \ldots, d_{N-1} \rangle$. Also, we denote as $\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle$ the *projection* of $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ over $\langle d_0, d_1, \ldots, d_{N-1} \rangle$. It should be noted that $\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle$ *contains precise multidimensional data stream readings* of the view V, being the latter defined at the leaf levels of hierarchies of the dimensions of V.

To give an example, Fig. 3 sketches an overview of the process that project a (imprecise) reading range (grey circles in Fig. 3) at level P_L over the (precise) reading base-range (red circles in Fig. 3) at level P_M. As shown in Fig. 3, the resulting range is then processed by the algorithm for computing possible-worlds probabilities.

Let $p^{PW}_{r,l}$ denote the possible-worlds probability to be assigned to the imprecise multidimensional data stream reading $r_{s,l}$. Since we deal with uncertain and imprecise data streams, $p^{PW}_{r,l}$ can be reasonably computed in terms of a *fraction* of the probability $p_{s,l}$ associated to $r_{s,l}$, as follows:

$$p^{PW}_{r,l} = \frac{p_{r,l}}{F} \tag{3}$$

such that $F > 0$ is an integer parameter. According to guidelines provided in Sect. 3, $p^{PW}_{r,l}$ captures *the likelihood of* $r_{s,l}$ *of being a precise reading within* $\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle$.

Before introducing our approach for computing $p^{PW}_{r,l}$, the definition of *neighborhood* for an imprecise reading $r_{s,l}$, denoted by $\mathcal{N}(r_{s,l})$, is necessary. Let $r_{s,l}$ be an imprecise reading, $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ the multidimensional range associated to $r_{s,l}$, and Δ a positive integer parameter (i.e., $\Delta > 0$). Since $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ is an n-dimensional domain, $r_{s,l}$ can be represented as an n-dimensional object in an

n-dimensional space, as follows: $r_{s,l} = \left\langle r_{s,l_{k_0}}, r_{s,l_{k_1}}, ..., r_{s,l_{k_{n-1}}} \right\rangle$. Let $r^L_{s,l}$ denotes the projection of $r_{s,l}$ over $\left\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle$. The Δ-*based neighborhood* of $r_{s,l}$ in $\left\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle$, denoted by $\Delta\mathcal{N}_{\left\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle}(r_{s,l})$, is defined as the set of *precise readings* contained in $\left\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle$ whose *Euclidean distance* from $r^L_{s,l}$ is lower or equal to Δ. Formally:

$$\Delta\mathcal{N}_{\left\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle}(r_{s,l}) = \left\{ \begin{array}{ll} r'_{s,l} & \left| r'_{s,l} \in \left\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle \wedge \right. \\ & r^L_{s,l} = \prod_{\left\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle}(r_{s,l}) \wedge \\ & \left. \sqrt{\sum_{i=0}^{n-1} \left(r'_{s,l_{k_i}} - r^L_{s,l_{k_i}} \right)^2} \leq \Delta \right. \end{array} \right\} \quad (4)$$

such that $\prod_{\mathcal{D}}(p)$ denotes the *projection operator* of a multidimensional point p over a multidimensional domain $\mathcal{D}$.

Following definition (4), we slightly modify the concept of possible-worlds probability $p^{PW}_{r,l}$, and introduce a variant that takes into account the Δ-based neighborhood of $r_{s,l}$, thus achieving the definition of the so-called Δ-*based possible-worlds probability*, denoted by $\Delta p^{PW}_{r,l}$. As we will demonstrate next, the convergence between these two possible-worlds probability concepts is ensured by the fact that, when Δ = 1, then $p^{PW}_{r,l} \equiv \Delta p^{PW}_{r,l}$.

From Sect. 3, recall that a combinatory dependence between the depth of the OLAP hierarchical level of the multidimensional range $\left\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \right\rangle$ associated to $r_{s,l}$ and computing $p^{PW}_{r,l}$ exists. We initially ignore this dependence and provide our proposed solution for computing $p^{PW}_{r,l}$ for the simplest case in which the multidimensional model of the view V is characterized by two (OLAP) levels only. This means that $\left\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \right\rangle$ is *directly contained* by $\langle d_0, d_1, \ldots, d_{N-1} \rangle$. In other words, the property above is described by the following constraint: $P_M = P_L + 1$. After describing the proposed solution for the baseline case $P_M = P_L + 1$, we generalize this solution for OLAP views exposing an arbitrary number of (OLAP) levels.

3.1 Baseline Case $P_M = P_L + 1$

Given an imprecise reading $r_{s,l}$ with multidimensional range $\left\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \right\rangle$ and a precise reading $r'_{s,l}$ contained in $\left\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \right\rangle$, we define the *probabilistic confidence interval distance* (PCID) of $r_{s,l}$ with respect to $r'_{s,l}$, denoted by PCID($r_{s,l}$, $r'_{s,l}$), as follows:

$$PCID(r_{s,l}, r'_{s,l}) = \frac{|v_{s,l,max} - v'_{s,l,max}|}{|v_{s,l,min} - v'_{s,l,min}|} \cdot p'_{s,l} \quad (5)$$

Intuitively enough, PCID($r_{s,l}$, $r'_{s,l}$) is a *probabilistic factor* modeling how much the confidence interval of $r_{s,l}$ is "*distant*" from the confidence interval of $r'_{s,l}$. It should be noted that the less the quantity $\frac{|v_{s,l,max} - v'_{s,l,max}|}{|v_{s,l,min} - v'_{s,l,min}|}$ the less the absolute distance between the confidence interval of $r_{s,l}$ and the confidence interval of $r'_{s,l}$. Also, since the confidence interval [$v'_{s,l,min}$, $v'_{s,l,max}$] of $r'_{s,l}$ is a *probabilistic estimate itself*, such that $p'_{s,l}$ is the probability associated to it, the quantity $\frac{|v_{s,l,max} - v'_{s,l,max}|}{|v_{s,l,min} - v'_{s,l,min}|}$ must be multiplied by $p'_{s,l}$. This determines definition (5), and allows us to capture the "reliability" of PCID($r_{s,l}$, $r'_{s,l}$) in modeling the probabilistic distance between the confidence interval of $r_{s,l}$, [$v_{s,l,min}$, $v_{s,l,max}$], and the confidence interval of $r'_{s,l}$, [$v'_{s,l,min}$, $v'_{s,l,max}$]. Linearly, a low absolute distance involves in a low probabilistic distance, and vice-versa.

Given an imprecise reading $r_{s,l}$ with multidimensional range $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$ and its Δ-based neighborhood in $\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle$, $\Delta\mathcal{N}_{\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle}(r_{s,l})$, we define the *Δ-based neighborhood probabilistic confidence interval distance* of $r_{s,l}$ with respect to $\Delta\mathcal{N}_{\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle}(r_{s,l})$, denoted by $\Delta\mathcal{N}PCID\left(r_{s,l}, \langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle\right)$, as follows:

$$\Delta\mathcal{N}PCID\left(r_{s,l}, \langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle\right) = \frac{\sum_{r'_{s,l} \in \Delta\mathcal{N}_{\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle}(r_{s,l})} PCID(r_{s,l}, r'_{s,l})}{\mathsf{COUNT}_p(\Delta\mathcal{N}_{\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle}(r_{s,l}))} \quad (6)$$

such that $\mathsf{COUNT}_p\left(\Delta\mathcal{N}_{\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle}(r_{s,l})\right)$ is an aggregate function that returns the number of precise readings in $\langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle$.

Intuitively enough, $\Delta\mathcal{N}PCID\left(r_{s,l}, \langle a^{L_M}_{s,l,k_0}, a^{L_M}_{s,l,k_1}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle\right)$ is a *probabilistic factor* modeling how much the confidence interval of $r_{s,l}$ is "distant" from confidence intervals of precise readings in the Δ-based neighborhood $\Delta\mathcal{N}_{\langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle}(r_{s,l})$ (of $r_{s,l}$).

Upon the theoretical framework above, given an imprecise reading $r_{s,l}$ with multidimensional range $\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \rangle$, the Δ-based possible-worlds probability of $r_{s,l}$, $\Delta p^{PW}_{r,l}$, is obtained as follows:

$$\Delta p^{PW}_{r,l} = \frac{p_{s,l}}{\Delta\mathcal{N}PCID\left(r_{s,l}, \langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle\right)} \cdot \frac{1}{\sum_{r_{s,l} \in \langle a^{L_M}_{s,l,k_0}, ..., a^{L_M}_{s,l,k_{n-1}} \rangle} \Delta p^{PW}_{r,l}} \quad (7)$$

Intuitively enough, $\Delta p_{r,l}^{PW}$ is obtained in terms of a fraction of the probability $p_{s,l}$ associated to $r_{s,l}$, where the denominator is represented by the probabilistic factor $\Delta\mathcal{N}$PCID that *globally* takes into account the "distance" of the confidence interval of $r_{s,l}$ from confidence intervals of their neighboring precise readings in $\Delta\mathcal{N}_{\left\langle a_{s,l,k_0}^{L_M},\ldots,a_{s,l,k_{n-1}}^{L_M}\right\rangle}(r_{s,l})$. Also, $\Delta p_{r,l}^{PW}$ is normalized with respect to all the other possible-worlds probabilities of imprecise readings in $\Delta\mathcal{N}_{\left\langle a_{s,l,k_0}^{L_M},\ldots,a_{s,l,k_{n-1}}^{L_M}\right\rangle}(r_{s,l})$, in order to achieve a full probabilistic interpretation of the introduced theoretical setting.

When $\Delta = 1$, we revise definition (7) as follows:

$$p_{r,l}^{PW} = \frac{p_{s,l}}{\Delta\mathcal{N}PCID\left(r_{s,l},\left\langle a_{s,l,k_0}^{L_M},\ldots,a_{s,l,k_{n-1}}^{L_M}\right\rangle\right)\Big|_{\Delta=1}} \cdot \frac{1}{\sum_{r_{s,j}\in\left\langle a_{s,l,k_0}^{L_M},\ldots,a_{s,l,k_{n-1}}^{L_M}\right\rangle} p_{r,l}^{PW}} \tag{8}$$

To better understanding, Fig. 4 sketches a conceptual example of our proposed approach. In the left side of Fig. 4, red circles represent the imprecise readings for which possible-worlds probabilities are computed on the basis of their "distance" from the precise readings (grey circles). This methodology is conceptually equivalent to selecting a "suitable" possible world from the collection of available ones (right side of Fig. 4).

3.2 Generalized Case $P_M > P_L + 1$

Now focus the attention on the *extended* model for computing the possible-worlds probability of a given imprecise multidimensional data stream reading $r_{s,l}$ in the case in which the multidimensional range $\left\langle a_{s,l,k_0}, a_{s,l,k_1},\ldots,a_{s,l,k_{n-1}}\right\rangle$ associated to $r_{s,l}$ is defined via combining n dimensions of the multidimensional model of V at *arbitrary* levels of their respective hierarchies, among those available in the whole OLAP hierarchy of V. For the sake of simplicity, assume that depths of these levels are equal for *all* the N hierarchies of V. Recall that P_L denotes this common depth, and P_M the depth of $\langle d_0, d_1, \ldots, d_{N-1}\rangle$ in the OLAP hierarchy of V, respectively. Therefore, $P_M - P_L$ levels exist between $\left\langle a_{s,l,k_0}, a_{s,l,k_1},\ldots,a_{s,l,k_{n-1}}\right\rangle$ and $\langle d_0, d_1, \ldots, d_{N-1}\rangle$ in the OLAP hierarchy of V.

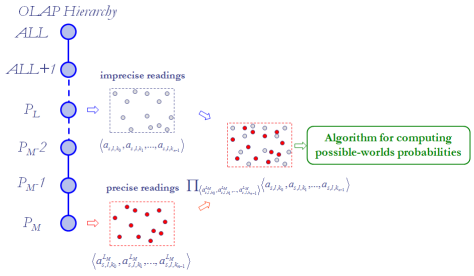


Fig. 3. Projecting a range of imprecise readings over the base range of precise readings

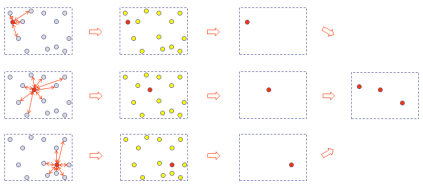


Fig. 4. Computing possible-worlds probabilities and selection of a "suitable" possible world

In order to compute the possible-worlds probability of $r_{s,l}$, $p_{r,l}^{PW}$, we simply *iterate* the baseline method given for a singleton pair of multidimensional ranges, i.e. $\left\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \right\rangle$ and $\langle d_0, d_1, \ldots, d_{N-1} \rangle$ of the previous case $P_M = P_L + 1$ (see Sect. 4.1), for *each* pair of multidimensional ranges defined at two *consecutive* levels of the OLAP hierarchy of V between P_L and P_M. In more detail, we pick pairs of multidimensional ranges across the OLAP hierarchy of V, starting from $\left\langle a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{n-1}} \right\rangle$ at level P_L and until $\langle d_0, d_1, \ldots, d_{N-1} \rangle$ at level P_M is reached. Each pair of multidimensional ranges is constituted by the *actual* multidimensional range at level L_k, $\left\langle a_{s,l,k_0}^{L_k}, a_{s,l,k_1}^{L_k}, ..., a_{s,l,k_{n-1}}^{L_k} \right\rangle$, and its *projection* over the multidimensional range at the underlying level L_{k+1}, $\left\langle a_{s,l,k_0}^{L_{k+1}}, a_{s,l,k_1}^{L_{k+1}}, ..., a_{s,l,k_{n-1}}^{L_{k+1}} \right\rangle$. Therefore, for each pair of multidimensional ranges $\left\langle a_{s,l,k_0}^{L_k}, a_{s,l,k_1}^{L_k}, ..., a_{s,l,k_{n-1}}^{L_k} \right\rangle$ and $\left\langle a_{s,l,k_0}^{L_{k+1}}, a_{s,l,k_1}^{L_{k+1}}, ..., a_{s,l,k_{n-1}}^{L_{k+1}} \right\rangle$, a possible-worlds probability $p_{r,l_{k,k+1}}^{PW}$ is obtained. The final possible-worlds probability associated to $r_{s,l}$, $p_{r,l}^{PW}$, is obtained by multiplying *all* these probabilities, as follows:

$$p_{r,l}^{PW} = \prod_{k=P_L}^{P_M - 1} p_{r,l}^{PW}{}_{k,k+1} \tag{9}$$

4 Supporting OLAP Query Evaluation over Uncertain and Imprecise Multidimensional Data Streams via Reliable Probabilistic Estimators

In this Section, we first provide our proposed innovative $\langle \varepsilon, \delta \rangle$-*based probabilistic estimator* [20] that retrieves estimates to OLAP queries via *appropriate statistics* extracted from PDF describing the uncertain and imprecise multidimensional data stream readings, based on the probabilistic models and tools presented in Sect. 3 and Sect. 4. Upon this conceptual basis, we then provide description and fundamental features of algorithms for supporting OLAP query evaluation.

4.1 A ⟨ε, δ⟩-Based Probabilistic Estimator for OLAP Queries over Uncertain and Imprecise Multidimensional Data Streams

Given a set of multiple uncertain and imprecise multidimensional data stream readings populating the reference M-dimensional OLAP view V and a fixed input OLAP query Q over V, our final goal is to provide an accurate estimate to Q, $\tilde{Q}$. Q can be modeled as follows:

$$Q = \left\langle \mathcal{A}, R_{k_0}, R_{k_1}, ..., R_{k_{m-1}} \right\rangle \tag{10}$$

The OLAP query definition (10) can be slightly modified in order to achieve the definition of a *continuous OLAP query* over V, denoted as $Q(t) = \langle Q, t \rangle$, such that Q is

a conventional OLAP query over V (defined like in the previous case) and t is a timestamp in which the fixed query Q is evaluated against V. By iterating the query task above for a set of timestamps $T_Q = \{t_q, t_{q+1}, \ldots, t_{q+k}\}$, we finally obtain a set of queries $Q(t) = \{Q(t_q), Q(t_{q+1}), \ldots, Q(t_{q+k})\}$ and a set of estimates $\tilde{Q}(t) = \{\tilde{Q}(t_q), \tilde{Q}(t_{q+1}), \ldots, \tilde{Q}(t_{q+k})\}$, such that $\tilde{Q}(t_q)$ is the estimate to the query $Q(t_q) = \langle Q, t_q \rangle$ at timestamp t_q. $\tilde{Q}(t)$ represents the continuous answer to the continuous query $Q(t)$.

For the sake of simplicity, in the following we focus the attention on the baseline problem of providing the accurate estimate $\tilde{Q}$, due to the fact that providing the continuous answer $\tilde{Q}(t)$ to the continuous query $Q(t)$ can be straightforwardly obtained via simply iterating the baseline solution.

From Sect. 1, recall that, similarly to state-of-the-art data stream query processing techniques, in our framework we make use of a buffer b of size B for query efficiency purposes. Therefore, in our reference application scenario (see Sect. 1), consumer applications are interested in executing OLAP queries over the collection of data stream readings stored within b.

Due to uncertainty an imprecision of multidimensional data stream readings, a probabilistic estimator $\Psi(Q)$ must be introduced in order to provide an accurate estimate to Q, $\tilde{Q}$. To this end, our general approach consists in providing $\tilde{Q}$ in terms of *some statistics* extracted from an *appropriate* PDF describing uncertain and imprecise multidimensional data stream readings [7,18]. Thanks to the probabilistic data stream model introduced in Sect. 2 and the possible-world semantics for OLAP over uncertain and imprecise multidimensional data streams introduced in Sect. 3, *we achieve the definition of a reliable $\langle \varepsilon, \delta \rangle$-based probabilistic estimator over uncertain and imprecise multidimensional data streams*, denoted by $\langle \varepsilon, \delta \rangle$-$\Psi(Q)$. Therefore, since the theory for probabilistic estimators is already available and it can be directly exploited within our proposed framework for OLAP over uncertain and imprecise multidimensional data streams, the most important research contribution we provide in this respect is represented by the *methodology for building the appropriate PDF describing uncertain and imprecise multidimensional data stream readings.* We next discuss this aspect, which plays a critical role in our research.

First, note that, in classical probabilistic estimators like the Hoeffding-based one [17], the PDF describing the set of target random variables $\mathcal{X}_p$, denoted by P_p, *is not available*, and, in fact, the underlying goal of the Hoeffding's inequality just consists in "re-constructing" this unknown PDF in order to retrieve probabilistic bounds over estimates of observed parameters. Now focus the attention on probabilistic properties of our proposed framework for OLAP over uncertain and imprecise multidimensional data streams. For each precise data stream reading $r'_{s,l}$, thanks to our probabilistic data stream model (see Sect. 2), the B-bounded discrete PDF describing the variation of possible values of $r'_{s,l}$ in terms of confidence intervals, $P^B_{r'_{s,l}}$, is available. For each imprecise data stream reading $r_{s,l}$, thanks to our possible-world semantics for OLAP over uncertain and imprecise data stream (see Sect. 3), a B-bounded discrete PDF accomplishing the same task illustrated for the case of *precise* data stream readings,

$P^B_{r_{s,l}}$, can be computed. This allows us to achieve a *unifying approach* for treating both precise and imprecise data stream readings, respectively, which can be both described by means of nice discrete PDF. Therefore, for the sake of simplicity, in the remainder of this paper we simple refer to multidimensional data stream readings $r_{s,l}$ (for both precise and imprecise instances) and related PDF, $P^B_{r_{s,l}}$.

4.2 Retrieving Estimates to OLAP Queries over Uncertain and Imprecise Multidimensional Data Streams

The nice unifying PDF-based probabilistic data model presented in Sect. 4.2 allows us to *directly retrieve* an accurate estimate $\tilde{Q}$ to an input OLAP query Q over the multidimensional view V via appropriate statistics extracted from the PDF describing the involved uncertain and imprecise multidimensional data stream readings. This is a significant contribution over the methodology proposed in [4], which is tailored to static OLAP data only.

For the sake of simplicity, consider SUM-based OLAP queries over uncertain and imprecise multidimensional data streams as the reference class of queries we deal with. Note that other kinds of OLAP queries (such as COUNT-based, AVG-based etc) can be easily obtained on top of SUM-based ones.

Given a SUM-based OLAP query $Q = \langle \text{SUM}, R_{k_0}, R_{k_1}, ..., R_{k_{m-1}} \rangle$ over the M-dimensional OLAP view V populated by precise and imprecise multidimensional data stream readings, our approach for providing a theoretically-founded estimate to Q, $\tilde{Q}$, is based on the following 3-step methodology: (1) On the basis of the multidimensional selectivity of Q, obtained by combining the dimensional ranges $R_{k_0}, R_{k_1}, ..., R_{k_{m-1}}$ into the multidimensional range $\langle R_{k_0}, R_{k_1}, ..., R_{k_{m-1}} \rangle$, retrieve the set of J PDF associated to data stream readings of V involved by Q, denoted by $\mathcal{P}^B_Q = \{P^B_{r_{s,l,0}}, P^B_{r_{s,l,1}}, ..., P^B_{r_{s,l,J-1}}\}$ — this can be simply performed via checking, for each reading $r_{s,l}$, if *all* its dimensional members $a_{s,l,k_0}, a_{s,l,k_1}, ..., a_{s,l,k_{m-1}}$ (see Sect. 2) are contained within $\langle R_{k_0}, R_{k_1}, ..., R_{k_{m-1}} \rangle$, i.e. $a_{s,l,k_j} \subset \langle R_{k_0}, R_{k_1}, ..., R_{k_{m-1}} \rangle$ for each k_j in {0, 1, …, M-1}. (2) From the set of PDF $\mathcal{P}^B_Q$, compute the *joint PDF* [20], denoted by $\mathcal{G}^B_Q$ — intuitively enough, $\mathcal{G}^B_Q$ models the joint contribution of *all* the PDF associated to data stream readings involved by Q. (3) Retrieve an accurate estimate to Q, $\tilde{Q}$, as follows:

$$Q = \langle \mathcal{A}, R_{k_0}, R_{k_1}, ..., R_{k_{m-1}} \rangle \tag{11}$$

Since step 1 is trivial and it does not deserve particular attention, we next explode and discuss steps 2 and 3, which, indeed, are more significant than the step 1.

Step 2 computes the joint PDF from the PDF set $\mathcal{P}_Q^B$, $\mathcal{G}_Q^B$. The latter is the main contribution of our research regarding to querying uncertain and imprecise data streams. To this end, we propose an approach inspired by [21], where PDF describing *independent observations* are combined for *pre-aggregation* purposes in the context of *probabilistic Data Warehouse servers*. According to our proposed approach, given two PDF $P_{r_{s,l_i}}^B$ and $P_{r_{s,l_j}}^B$ describing the data stream readings r_{s,l_i} and r_{s,l_j}, respectively, the joint PDF $\mathcal{G}_{i,j}^B$ can be obtained as follows. For each component *k* of $\mathcal{G}_{i,j}^B$, the pair of confidence intervals $[v_{s,l,min_i}, v_{s,l,max_i}]$ and $[v_{s,l,min_j}, v_{s,l,max_j}]$ of $P_{r_{s,l_i}}^B[k]$ and $P_{r_{s,l_j}}^B[k]$, respectively (picked at the *same* buffer entry *k*), are used to compute the *new* confidence interval $[v_{s,l,min_i} + v_{s,l,min_j}, v_{s,l,max_i} + v_{s,l,max_j}]$. Similarly, the pair of probabilities p_{s,l_i} and p_{s,l_j} of $P_{r_{s,l_i}}^B[k]$ and $P_{r_{s,l_j}}^B[k]$, respectively (picked at the *same* buffer entry *k*), are then multiplied in order to obtain the *new* probability $p_{s,l_i} \cdot p_{s,l_j}$. The *k*-th component of $\mathcal{G}_{i,j}^B$ is finally obtained as follows:

$$\mathcal{G}_{i,j}^B[k] = \left\langle [v_{s,l,min_i} + v_{s,l,min_j}, v_{s,l,max_i} + v_{s,l,max_j}], p_{s,l_i} \cdot p_{s,l_j} \right\rangle \tag{12}$$

Step 3 retrieves an accurate estimate to *Q*, $\tilde{Q}$, by means of the *multidimensional integral* (11), which models a conceptual formula showing how $\tilde{Q}$ can be retrieved, at a theoretical level, in the *continuous domain*. Recall that we handle SUM-based OLAP queries over uncertain and imprecise multidimensional data stream readings, so that the integral operator (10) well-defines $\tilde{Q}$ as the summation of partial contributions over the continuous domain. Since we focus on discrete PDF (see Sect. 2), (11) is then specialized for the *discrete domain*, as follows:

$$\tilde{Q} = \sum_{\langle k_0, k_1, \ldots, k_{m-1} \rangle = \langle R_{k_0}[0], R_{k_1}[0], \ldots, R_{k_{m-1}}[0] \rangle}^{\langle R_{k_0}[|R_{k_0}|-1], R_{k_1}[|R_{k_1}|-1], \ldots, R_{k_{m-1}}[|R_{k_{m-1}}|-1] \rangle} \mathcal{G}_Q^B[\langle k_0, k_1, \ldots, k_{m-1} \rangle] \tag{13}$$

such that (*i*) $\langle k_0, k_1, \ldots, k_{m-1} \rangle$ denotes an *m*-dimensional point, (*ii*) $R_{k_i}[h]$ the *h*-th dimensional member of the dimensional range R_{k_i}, and (*iii*) $|R_{k_i}|$ the cardinality of R_{k_i}.

Furthermore, since (*i*) $\mathcal{G}_Q^B$ globally models the joint contribution of all the PDF associated to data stream readings of *V* involved by *Q*, and (*ii*) the multidimensional selection performed at step 1 of our OLAP query estimation technique selects *only* the set of *J* PDF associated to the involved readings, we can *break* the dependency of $\mathcal{G}_Q^B$ from the *m*-dimensional component exposed in (11), and retrieve $\tilde{Q}$ by means of computing appropriate statistics over $\mathcal{G}_Q^B$. From the literature, relevant *moments* [20] of $\mathcal{G}_Q^B$, i.e. *mean*, denoted by $\mathbf{E}(\mathcal{G}_Q^B)$, and *variance*, denoted by $\mathbf{V}(\mathcal{G}_Q^B)$, play the role

of most appropriate statistics in this respect. In fact, according to several studies [7,18], these moments well-summarize the *statistical content* of $\mathcal{G}_Q^B$, and are well-suited for OLAP queries over the involved (uncertain and imprecise) readings. For instance, the estimate to a SUM-based OLAP query can be obtained as follows:

$$\tilde{Q} = B \cdot \mathbf{E}(\mathcal{G}_Q^B) \tag{14}$$

where (*i*) B denotes the buffer size (see Sect. 1), and (*ii*) $\mathbf{E}(\mathcal{G}_Q^B)$ the mean of $\mathcal{G}_Q^B$.

Given a discrete PDF $P[k]$ with q samples, the mean $\mathbf{E}(P[k])$ of $P[k]$ is defined as follows [20]:

$$\mathbf{E}(P[k]) = \sum_{i=0}^{q-1} k_i \cdot P[k_i] \tag{15}$$

whereas the variance $\mathbf{V}(P[k])$ of $P[k]$ is defined as follows [20]:

$$\mathbf{V}(P[k]) = \mathbf{E}(P[(k-\mu)^2]) = \sum_{i=0}^{q-1} (k_i - \mu)^2 \cdot P[k_i] \tag{16}$$

5 Experimental Analysis and Results

In order to test the performance of our proposed framework for estimating OLAP queries over uncertain and imprecise multidimensional data streams, we conducted an experimental analysis on several classes of synthetic multidimensional data stream sets. The goal of our analysis consists in evaluating the *accuracy* of retrieved estimates under the ranging of the following critical parameters: (*i*) the percentage of imprecise readings contained by the multidimensional view exploited in the target OLAP analysis; (*ii*) the dimensionality of input uncertain and imprecise multidimensional data streams (this also tests the *scalability* of our proposed framework).

As regards the data layer of our experimental framework, we considered three classes of synthetic data stream sets whose reading values follow three distinct data distributions: *Uniform*, *Gauss* and *Zipf*. It should be noted that these data stream sets well-describe popular cases one can find in real-life data stream settings. In addition to this, since our proposed possible-worlds semantics for OLAP over uncertain and imprecise multidimensional data streams is data-driven in nature, ranging reading values according to different data distributions represents a reliable way of testing the effective capabilities (and limitations) of our proposed framework in suitable close-to-similar operational settings. On the other hand, one of the advantages of synthetic data streams over real-life ones relies in the fact that the experimental framework built on top of such data stream sets can be easily customizable in order to stress several aspects and experimental parameters/behaviors of the target data stream processing algorithm/technique.

Looking into details, (*i*) the Uniform data stream set has been generated according to a Uniform distribution ranging on the interval [50, 100], (*ii*) the Gauss data stream set has been generated according to a *normal* Gauss distribution, while (*iii*) the Zipf

data stream set has been generated according to a Zipfian distribution whose parameter z ranges on the interval [0.5, 1.0]. For each synthetic data stream set, uncertainty of OLAP measures has been captured by means of *randomly-generated* confidence intervals over actual synthetic data stream reading values.

As regards the query layer of our experimental framework, we considered a fixed OLAP range-SUM query whose selectivity was equal to the 30 % of the whole selectivity of the (output) OLAP view. Recall that in multidimensional spaces defined by data cubes, views and OLAP queries, selectivity can be simply intended in terms of the *volume* of the respective objects.

Finally, as regards the metrics of our experimental framework, we considered the *percentage relative query error* ε_Q of the estimate to the (fixed) input range-SUM query Q, via comparing the estimate $\tilde{Q}$ against the exact answer $\hat{Q}$ (known in advance thanks to the completely-controlled synthetic data stream generator). Formally, ε_Q is defined as follows:

$$\varepsilon_Q = \frac{|\hat{Q} - \tilde{Q}|}{\hat{Q}} \tag{17}$$

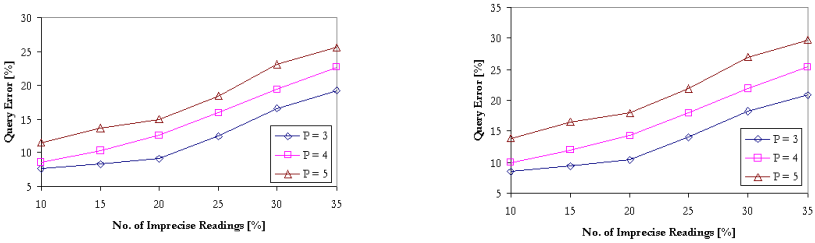
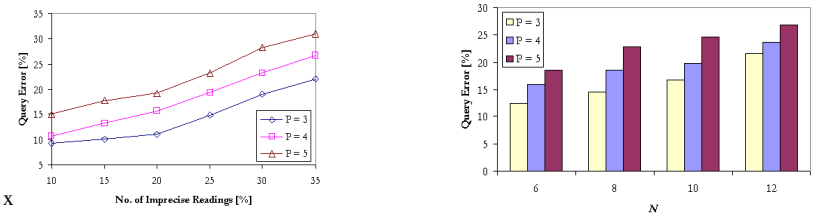


Fig. 5. ε_Q vs *NoIR* on a 6D Uniform (*a*) and a 6D Gauss (*a*) data stream set ($M = 4$)

In the first experiment, we considered 6-dimensional synthetic data stream sets (i.e., $N = 6$) whose number of (OLAP) levels varies on the set {3, 4, 5}. The dimensionality of the target (output) OLAP view has been set to 4 (i.e., $M = 4$). In order to capture imprecision of multidimensional data streams, we simply considered the percentage number of imprecise readings (contained by the view), denoted by *NoIR*, with respect to the total number of precise readings that can be stored by the full view. Fig. 5 and Fig. 6 (*a*) show our experimental results obtained in the first experiment for Uniform, Gauss and Zipf 6-dimensional synthetic data stream sets, respectively, with respect to the percentage number of imprecise readings *NoIR* when ranging the number of OLAP levels P of input data streams ($M = 4$).

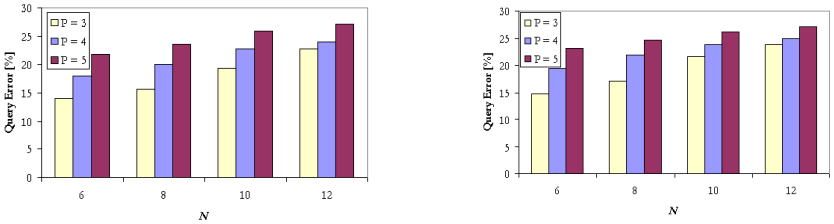
In the second experiment, we stressed the scalability of our proposed framework by ranging the number of dimension of input data streams. The percentage number of imprecise readings has been set to 25 % (i.e., *NoIR* = 25 %). Number of OLAP levels varies again on the set {3, 4, 5}. Fig. 6 (*b*) and Fig. 7 show our experimental results obtained in the second experiment for Uniform, Gauss and Zipf multidimensional

synthetic data stream sets, respectively, with respect to the number of dimensions of input data streams N when ranging the number of OLAP levels P of input data streams ($NoIR$ = 25%).



(*a*) (*b*)

Fig. 6. ε_Q vs $NoIR$ on a 6D Zipf data stream set (M = 4) (*a*) and ε_Q vs N on Uniform data stream sets (M = 4) (*b*)



(*a*) (*b*)

Fig. 7. ε_Q vs N on Gauss (*a*) and Zipf (*b*) data stream sets (M = 4)

Figg. 5-7 confirm to us the benefits of our proposed framework for estimating OLAP queries over uncertain and imprecise multidimensional data streams, which can be summarized as follows. First, retrieved query errors are perfectly tolerable and follow canonical thresholds of state-of-the-art approximate query answering techniques in OLAP (e.g., see [4,6,18]). Second, although being data-driven in nature, our framework results to be marginally dependent on the data distribution of input multidimensional data stream readings. The latter is a nice feature, as real-life data streams expose heterogeneous distributions. Third, performance decreases when the number of OLAP levels of input multidimensional data streams increases, as expected. This because our data-driven method for computing probabilities of imprecise multidimensional data stream readings retrieves less accurate probabilities as the number of OLAP levels increases, due to its combinatory dependence on the latter parameters and the fact that, when a number of OLAP levels greater than 2 is considered, actual probabilities are obtained by multiplying (already-)derived probabilities, so that the inaccuracy of estimates is “propagated” and “magnified” across OLAP levels. Finally, in addition to properties above, retrieved query errors are even lowly dependent on the dimensionality of input data streams, which is a

critical aspect affecting the scalability of the proposed framework in real-life data multidimensional stream settings.

6 Conclusions and Future Work

In this paper, we have proposed a novel framework for estimating OLAP queries over uncertain and imprecise multidimensional data streams. The proposed framework introduces some relevant research contributions, and the suitability of the framework in the context of modern data stream applications and systems, which are more and more characterized by the presence of uncertainty and imprecision, has been demonstrated throughout a campaign of experiments on several classes of synthetic multidimensional data stream sets. Future work is mainly oriented towards making our proposed framework robust with respect to complex OLAP aggregations over uncertain and imprecise multidimensional data streams, according to research directions described in [13], and beyond simple SQL-based OLAP aggregations (e.g., SUM, COUNT etc) like those investigated in this paper.

References

1. Abadi, D., Carney, D., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: A New Model and Architecture for Data Stream Management. VLDB Journal 12(2) (2003)
2. Aggarwal, C.C., Yu, P.S.: A Framework for Clustering Uncertain Data Streams. In: IEEE ICDE (2008)
3. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: ACM PODS (2002)
4. Burdick, D., Deshpande, P., Jayram, T.S., Ramakrishnan, R., Vaithyanathan, S.: OLAP over Uncertain and Imprecise Data. In: VLDB (2005)
5. Cai, Y.D., Clutterx, D., Papex, G., Han, J., Welgex, M., Auvilx, L.: MAIDS: Mining Alarming Incidents from Data Streams. In: ACM SIGMOD (2004)
6. Chen, Y., Dong, G., Han, J., Wah, B.W., Wang, J.: Multi-Dimensional Regression Analysis of Time-Series Data Streams. In: VLDB (2002)
7. Cormode, G., Garofalakis, M.: Sketching Probabilistic Data Streams. In: ACM SIGMOD (2007)
8. Cormode, G., Korn, F., Tirthapura, S.: Exponentially Decayed Aggregates on Data Streams. In: IEEE ICDE (2008)
9. Cuzzocrea, A., Chakravarthy, S.: Event-based Lossy Compression For Effective And Efficient OLAP Over Data Streams. Data & Knowledge Engineering 69(7) (2010)
10. Cuzzocrea, A., Furfaro, F., Masciari, E., Saccà, D.: Improving OLAP Analysis of Multidimensional Data Streams via Efficient Compression Techniques. In: Cuzzocrea, A. (ed.) Intelligent Techniques for Warehousing and Mining Sensor Network Data. IGI Global (2009)
11. Cuzzocrea, A., Furfaro, F., Mazzeo, G.M., Saccá, D.: A grid framework for approximate aggregate query answering on summarized sensor network readings. In: Meersman, R., Tari, Z., Corsaro, A. (eds.) OTM-WS 2004. LNCS, vol. 3292, pp. 144–153. Springer, Heidelberg (2004)

12. Cuzzocrea, A., Serafino, P.: LCS-Hist: Taming Massive High-Dimensional Data Cube Compression. In: EDBT (2009)
13. Dobra, A., Gehrke, J., Garofalakis, M., Rastogi, R.: Processing Complex Aggregate Queries over Data Streams. In: ACM SIGMOD (2002)
14. Dalvi, N.N., Suciu, D.: Efficient Query Evaluation on Probabilistic Databases. In: VLDB (2004)
15. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Data Mining and Knowledge Discovery 1(1) (1997)
16. Han, J., Chen, Y., Dong, G., Pei, J., Wah, B.W., Wang, J., Cai, Y.D.: Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams. Distributed and Parallel Databases 18(2) (2005)
17. Hellerstein, J.M., Haas, P.J., Wang, H.J.: Online Aggregation. In: ACM SIGMOD (1997)
18. Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating Statistical Aggregates on Probabilistic Data Streams. In: ACM PODS (2007)
19. Jin, C., Yi, K., Chen, L., Xu Yu, J., Lin, X.: Sliding-Window Top-K Queries on Uncertain Streams. PVLDB 1(1) (2008)
20. Papoulis, A.: Probability, Random Variables, and Stochastic Processes, 2nd edn. McGraw-Hill, New York City (1984)
21. Timko, I., Dyreson, C.E., Pedersen, T.B.: Pre-Aggregation with Probability Distributions. In: ACM DOLAP (2006)

Data Value Storage for Compressed Semi-structured Data

Brian G. Tripney[1], Isla Ross[1], Francis A. Wilson[2], and John N. Wilson[1]

[1] Department of Computer & Information Sciences,
University of Strathclyde, Glasgow, UK
[2] Graduate School of Business, University of the South Pacific, Suva, Fiji
{brian,jnw,isla}@cis.strath.ac.uk, wilson_f@usp.ac.fj

Abstract. Growing user expectations of anywhere, anytime access to information require new types of data representations to be considered. While semi-structured data is a common exchange format, its verbose nature makes files of this type too large to be transferred quickly, especially where only a small part of that data is required by the user. There is consequently a need to develop new models of data storage to support the sharing of small segments of semi-structured data since existing XML compressors require the transfer of the entire compressed structure as a single unit. This paper examines the potential for bisimilarity-based partitioning (i.e. the grouping of items with similar structural patterns) to be combined with dictionary compression methods to produce a data storage model that remains directly accessible for query processing whilst facilitating the sharing of individual data segments. Study of the effects of differing types of bisimilarity upon the storage of data values identified the use of both forwards and backwards bisimilarity as the most promising basis for a dictionary-compressed structure. A query strategy is detailed that takes advantage of the compressed structure to reduce the number of data segments that must be accessed (and therefore transferred) to answer a query. A method to remove redundancy within the data dictionaries is also described and shown to have a positive effect on memory usage.

1 Introduction

New directions in the provision of end-user computing experiences make it necessary to determine the best way to share online data. The volume of data available over the Internet grows on a daily basis. At the same time, end users' expectations are increasing, with smartphone users now expecting instant access to information wherever they may be. The array of different processing techniques in use necessitates a standard format for data exchange and the self-describing nature of semi-structured data, in particular XML, has led to its common usage for this purpose. The side effect of this property is that file sizes quickly become large, with a high proportion of this being contributed by the description of the file format. XML compression techniques have partly addressed this by reducing storage requirements at the significant expense of requiring additional

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 174–188, 2013.

processing to access the data contained within the compressed files. However, an entire data structure is often not required, with users typically only interested in a small subset of the data. In such cases it follows that only the parts of the data structure of interest to the user need be accessed by the query processing system. Where the data is appropriately partitioned, or broken into segments, such an approach can limit the volume of data to be processed. A similar effect can be seen with data transfer. By only transporting the data segments directly involved in answering a query, the overall communications bandwidth utilised is also reduced. The effect is multiplied where additional queries can be answered using the data segments already held. In a system allowing sharing between peers, the data can be sourced from another local device rather than the server - again there is benefit in only sharing the segments required to resolve the query. Existing XML physical models are either non-queryable ([1], [2], [3], [4]) or have other drawbacks, e.g. requiring large sections of data to be decompressed in order to access a single value ([5], [6], [7], [8], [9], [10], [11]). In all cases these existing storage models require the entire data structure to be transferred as a single unit to allow any access to the data contained within. To facilitate sharing, a data storage model should be able to partition the semi-structured data into segments and store these in a manner that maximises utilisation of storage space while still making the stored values easily accessible. Semi-structured data can be separated into segments of related data by a process based around bisimilarity [12][13] - where items with similar structural patterns are grouped together. Such segmentation forms the basis of a data storage model that allows individual segments to be shared and recombined as required. Support for queries can be maintained by utilising an independent method of compression for each data segment - i.e. the whole structure should not be required to access the data stored in any one of the segments. The contribution of this paper is to characterise the effect of alternative approaches to bisimilarity on partitioning XML data structures with a view to identifying the most promising approach. Partitioning in this way produces redundant dictionaries and we examine the potential for curbing this problem. Lastly we demonstrate improvements in query performance using partitioned, compressed data structures. Section 2 reviews the previous work in the areas of semi-structured data compression, indexing and structural summarisation. The experimental work is set out in Section 4 and the results are presented and discussed in Section 5.

2 Background

XML documents are fundamentally tree-based structures although it is possible to superimpose links across the tree. This makes it convenient to use a datagraph as a convenient abstract representation of a document. The datagraph provides a graphical representation of the document structure showing each individual XML element as a node within a graph and the structure of the document as edges connecting the nodes. This presents a starting point for approaches to partitioning and indexing XML structures. Work-load aware methods of partitioning

Table 1. Comparison of existing XML compressors

	Queryable	XML Schema	Backend compressor	Decompression unit	Inequality without decompression	Comment
XMill[1]	No	No	gzip	Full	N/A	User must specify groupings and compressors to achieve claimed level of compression.
XMLPPM[2]	No	No	PPM	Full	N/A	Statistical modelling allows better compression than default XMill, but maintaining four models is slow.
SCA[3]	No	Req.	gzip	Full	N/A	Slower and less effective than XMill. Requires XML schema.
XWRT[4]	No	No	gzip	Full	N/A	Can improve compression, but must fully adjust parameters.
XGrind[5]	Yes	Opt.	Huffman	Value	No	Compression requires two passes over document. Querying requires parsing of entire compressed document. Only exact matches can be found.
XPRESS[6]	Yes	No	Huffman/ dictionary	Value	No	Improves querying over XGrind - no need for linear parse of document. Compression still considerably worse than XMill.
QXT[7]	Yes	No	gzip	Container	No	Compressed size is design priority. Must unzip full container before any examination of transformed contents.
XQueC[8]	Yes	No	ALM/ Huffman	Value	Poss.	Prefers advance knowledge of query workload to choose compressors. Requires large auxiliary data structures to permit querying - must manage pointers to individual items within containers.
XQzip[9]	Yes	No	gzip/ dictionary	Block	No	Values held in blocks of 1000. Requires decompression of full block to access single value. Authors recognise this and attempt to compensate with buffer pool.
XCQ[10]	Yes	Req.	gzip	Block	Part.	Stores less structure so smaller compressed files, but requires schema to do so.
ISX[11]	Yes	Opt.	gzip	Block	No	Emphasis on traversal of structural part.

XML data provide benefits in distributed processing of native XML structures [14]. Other partitioning techniques seek to improve query performance by mapping XML into object hierarchies [15]. Path-base partitioning [16] and variations that are tuned to particular query patterns [17] have also been shown to provide significant benefits. Work on DataGuides [18] recognised the repetitive nature of datagraphs and exploited it to aid querying of schema-less semi-structured databases. By extracting only unique paths of nodes that exist within the datagraph, the DataGuide produced is a compact and accurate summarisation of the database structure offering useful information for query authors. Bisimilarity [12][13] exploits patterns in the types of nodes in the datagraph that are connected to each other. Establishing bisimilarity is a two stage process. First, the two nodes must be labelled the same, i.e. the two elements represented by the nodes must be of the same type. Secondly, the paths connected to the two nodes are examined to ensure that the labels of the ancestor nodes (via the incoming paths) are the same for each node and that the labels of the descendant nodes (via the outgoing paths) are the same for each node. A variety of approaches to bisimilarity are possible by varying the direction and length of the paths examined. The A(k)-index [12] is a family of indices created using different levels of backwards bisimilarity (`k`), i.e. it is the incoming paths that are considered for purposes of determining bisimilarity. Forwards bisimilarity

```
<ContactList>
  <Student>
    <Name>Person1Name</Name>
    <Contact>
      <Email>person1name@example.com</Email>
    </Contact>
  </Student>
  <StaffMember>
    <Name>Person2Name</Name>
    <Contact>
      <Email>person2name@example.com</Email>
    </Contact>
  </StaffMember>
  <StaffMember>
    <Name>Person3Name</Name>
    <Contact>
      <Telephone>07780858382</Telephone>
    </Contact>
  </StaffMember>
</ContactList>
```

Listing 1.1. XML Example

(i.e. the comparison of outgoing paths) can be used to exploit the sharing of common sub-trees [13]. Applying forward and backward bisimilarity alternately until a point is reached where no further changes are made to the node groupings, produces an F&B-Index, which is the smallest index that accurately covers all branching path queries (i.e. those where the query does not take a linear path through the index)[19] Limiting the levels of bisimilarity used (as is the case for backwards bisimilarity in the A(k)-Index above) and limiting the number of times the main F&B-Index computation is performed produces a more compact (j,k)-F+B-Index[1] at the expense of accuracy [20]. These methods aid the access to the XML data by supplementing or replacing the structural part of the document. They also provide a basis for approaches to XML compression methods that involve both the document structures and the data values contained within them.

A variety of approaches have been used to design XML compressors that represent both the datagraph structure and the values contained within it. Characterisitcs of key systems are summarised in Table 1. While the non-queryable methods require full decompression before any further processing may take place, the queryable methods require different levels of decompression to answer a query. This is related to the choice of backend compressor, with those methods using gzip having to decompress complete blocks or containers to access a single value, while the methods built upon dictionary or Huffman compression are able to directly access the individual required value, reducing the decompression workload. A desirable feature is the ability to evaluate inequalities without the need to decompress individual values. A partial version of this is found in XCQ which can make use of the maximum and minimum values stored in the block statistics signature to quickly rule out an entire block of values. Thereafter the matching blocks must be decompressed in their entirety to complete the query. In XQueC,

[1] Where `j` and `k` specify the levels of forwards and backwards bisimilarity respectively.

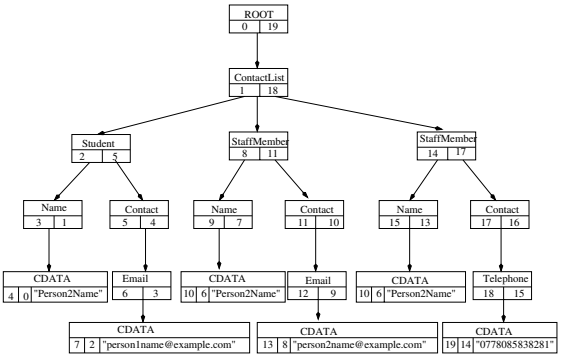


Fig. 1. Structural representation of forwards and backwards bisimilarity

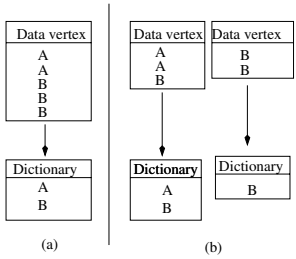


Fig. 2. Effect of data value distribution

the order-preserving nature of compression allows comparison of the individual compressed values. Additional user input in the form of advance knowledge of the query workloads is also required. This additional dependency on the user is also seen in XMill and XWRT, which require the user to manually set a number of parameters to achieve best performance.

The focus of the methods reviewed in Table 1 is the compression of entire data structures rather than methods that are designed to compress data structures that have been split into parts for sharing.

3 System Architecture

Bisimilarity-based structural summarisation naturally separates data into discrete groupings (partitions), which have the potential to form the basis of a system where small sections of a data structure can be distributed in an environment that requires data to be shared. XML data can be partitioned using a variety of approaches based around different charactersitics of the structure [21]. Bisimilarity takes the surrounding structure into account and is shown to provide a flexible method of grouping similar vertices, particularly in the context of real world data structures. The partitioning used is based on the F&B Index in which nodes of the datagraph are deemed to be bisimilar if their labels match and the same is true for both the ancestor nodes (incoming paths) and descendant nodes (outgoing paths) - this is applied repeatedly until a structure with stable node groupings is found. The structure is supplemented by a numbering scheme, which maintains the ordering of the data throughout [22]. Datagraph nodes are grouped by the bisimilarity function described earlier, so each entry within a particular vertex is of the one type and it is this type that appears at the top of each vertex in the diagram. Entries are marked with a pre-order number, which corresponds to the order in which the associated XML elements appear in the original document - thus maintaining the order of the stored data.

```
ROOT
0,19,1,21
ContactList
1,18,2,20
Student
2,5,3,7
Name
3,1,4,3
CDATA
4,0,4,1,''Person1Name''
Contact
5,4,4,4
Email
6,3,5,3
CDATA
7,2,5,1,''person1name@example.com
;;
.
.
.
```

Listing 1.2. NSIndex File Format

Table 2. Experimental data structures

	Regular	Irregular
Benchmark	Orders[1] Modified Orders[3]	XMark[2]
Real World	Legal[4]	NASA[5] Medline[6] Dream[7] Rat[8] Human[9]

[1]Subset of TPC-H represented as XML (15Mb)
[2]Variable structure and random text (10&30Mb)
[3]Value-based elements of TPC-H subset (15Mb)
[4]Court sentencing data (1&13Mb)
[5]Text of *Midsummer Night's Dream* (0.15Mb)
[6]US National Library of Medicine bib (20Mb)
[7]Astronomical Data centre bibliography (24Mb)
[8]Ensemble rat genome annotation data (25Mb)
[9]Ensemble human genome data (25Mb)

They are also marked with post-order number, obtained from the last time each element is encountered in the document. This may be thought of as the order in which the end tags are found - with data values also numbered. Also recorded in each vertex entry, though not shown in the diagram, is the level at which the entry appears within the structure and the size of the entry, including any sub-trees that appear beneath it in the structure. For example, the `Contact` entry `(17,16)` shown at the right-hand side of Figure 1 will have level 4 (`ROOT` is counted as level 1) and size 3, counting itself and the two entries below it in the structure - `Telephone(18,15)` and `CDATA(19,14,"07780858392")`. This example also shows that the structural part of the `Telephone` element is stored in vertex (`18,15`), while the data value itself is stored in a separate vertex that holds the data entry (`19,14,"07780858392"`).

The structure is extended by a storage module that allows the structure to be written to disk. The physical representation is comma separated values (CSV) format as shown by the example in Listing 1.2. The data type of the vertex is output first, followed by each entry contained within the vertex on subsequent lines. Each structural entry is recorded as a sequence of pre-order, post-order, level and size, with the data value additionally being recorded for data entries.

4 Experimental Work

The main computational challenges addressed by this work are to devise a consistent way of partitioning semi-structured data so that it is possible to represent the associated data values in a compact form and yet retain the ability to address each value separately in this compressed form without the need to decompress the entire data structure.

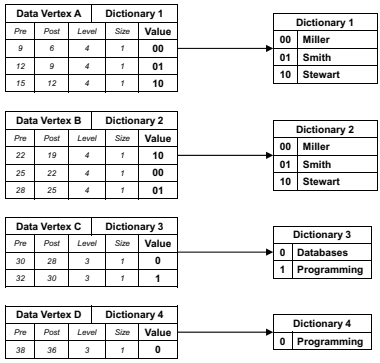


Fig. 3. Data vertices and dictionaries before reduction

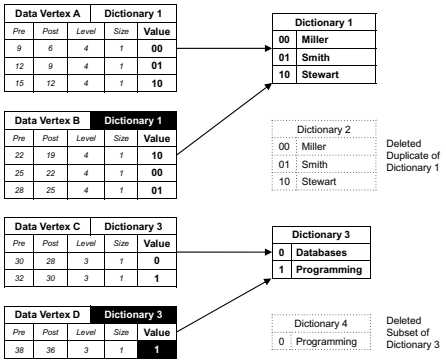


Fig. 4. Data vertices and dictionaries after reduction (updated values shaded)

To permit the sharing of independent segments of data it is necessary to make use of an appropriate storage method. This must arrange the data into sections and allow these to be transferred and accessed individually. While existing XML compression methods group data into containers for compression purposes, the entire compressed structure must be transferred before any querying may take place, making these methods unsuitable for sharing data segments independently. The method of data storage proposed here combines data partitioning with a system of compression to store the data values. Dictionary-based compression was chosen over character-based compression as a consequence of the overall functionality available with the former approach [23]. The structural summarisation produced by bisimilarity affects the groupings of data values. These experiments evaluate the effect of changing the partitioning scheme on the number and size of data dictionaries required. Since the distribution of data values influences dictionary structure, a range of real-world data sets were used to assess the effect of partitioning. These sets were further classified as being regular or irregular in structure and supplemented by benchmark data (Table 2)Regular benchmark data includes both value based data (ModifiedOrders) and a mix of value and text based (Orders).

Bisimilarity options (no bisimilarity where data entries are grouped by label only, full backwards bisimilarity, full forwards bisimilarity and full forwards and backwards bisimilarity) affect the compressed size of an XML structures partitioned by these approaches. Changing the partitioning has an effect on the number of data vertices over which the data values are distributed. It is this distribution of data values that has an effect on the overall compressed data size, as repeated data values within a single data vertex require only one unique entry in the associated data dictionary, while repeated values that occur across a number of data vertices will have an entry in multiple dictionaries. An example is given in Figure 2 using the data values `A`, `A`, `B`, `B`, `B`. If, as shown in Figure 2(a), these are partitioned in such a way that all of the `B` values fall into the same data vertex, then all three values are described by a single entry in the

```
ROOT
0,19,1,21
ContactList
1,18,2,20
Student
2,5,3,7
Name
3,1,4,3
CDATA, ContactList.0.dic        <-- dictionary reference
4,0,4,1,0                       <-- data value replaced by token
Contact
5,4,4,4
Email
6,3,5,3
CDATA, ContactList.1.dic        <-- dictionary reference
7,2,5,1,0                       <-- data value replaced by token
.
.
.
```

Listing 1.3. NSIndex Compressed File Format

dictionary associated with that data vertex. In this case the total dictionary size will be the size of value `A` plus the size of value `B`. In addition, each entry in the dictionary is related to an integer, which is used as a token to replace the entry in the datagraph structure. If as in Figure 2(b) the `B` values are separated into two separate data vertices, then an entry will be required in the dictionary associated with each data vertex containing a `B`. In this example, the result is that an extra dictionary entry for value `B` must be stored in the second data dictionary, increasing the overall dictionary size. This could have a considerable effect on the overall size of dictionaries, especially when longer data values are involved. A set of associated data dictionaries containing only the unique entries from each vertex was created for each of the four differently-partitioned sets of data vertices for each data set.

The overall size of the compressed data values and the associated dictionaries for each partitioning scheme is of interest here, in particular whether there are any particular bisimilarity options that perform consistently well across the various data sets used. Also of interest is the number of dictionaries produced as, with a view to creating manageable segments of data, the smaller the individual dictionaries, the more there will necessarily be. Listing 1.3 incorporates dictionary references and the encoded data values given in Listing 1.2.

When a datagraph structure is loaded from a file, each data vertex will hold the data entries with their tokenised data values and a reference to the associated dictionary. The dictionary itself is held centrally, which permits multiple vertices to refer to the same dictionary. Top-down queries over the datagraph would lead to a large number of dictionary decode actions and consequent poor query performance. Such an approach would select all data entries of the correct type for each query predicate before linking the entries that matched the last predicate back to the previous one and so on. An example is given in Table 3(a) based on the query `/A/B/"Data"`. Table 3(b) shows the compressed query strategy. At each stage, only those vertices that are children of the vertices at the previous

level and of the correct type are considered as potential results. This is in contrast to the uncompressed strategy, which noted all the individual data entries of the correct type for each stage regardless of where in the overall structure they were situated. The compressed strategy takes advantage of the entries being bundled together into vertices and these vertices being linked by edges. Traversing the edges means that only the child vertices are considered at the next stage of query resolution. This greatly reduces the number of vertices to be considered at each stage, which can be important when atomic values are being considered. For each data vertex to be checked, the query data value must be encoded using the appropriate dictionary before the contained data entries can be evaluated - it is therefore an advantage to have limited the number of data vertices to be checked.

Whichever partitioning method is chosen, there is potential for a large number of data dictionaries to be created. The nature of the bisimilarity-based summarisation is such that the logical domains will be split across a number of data vertices and there arises the possibility of repetition of values across the set of data dictionaries. This makes poor use of storage space, particularly where there are duplicate dictionaries but also where one dictionary is a wholly contained subset of another. Removing redundancy is straightforward in the case of duplicate dictionaries - the duplicate is deleted and any data vertices that referenced it are updated to make use of the remaining dictionary. There is no need to change the compressed data tokens contained in these data vertices. For subsets, the process is slightly more involved as the additional values contained within the superset dictionary mean that the values in data vertices previously using the subset dictionary may be represented by different tokens in the superset dictionary. Therefore to rationalise the dictionaries, a translation table must be created to update the old tokens in the data vertex, which refer to values in the subset dictionary, to tokens that point to the same value in the new superset dictionary. An example of these processes is shown in Figures 3 and 4. The compressed data vertices and dictionaries before the reduction process are illustrated in Figure 3 with each of the four example data vertices having an associated dictionary. Dictionaries 1 and 2 are duplicates of each other and the values stored in Dictionary 4 are a subset of those in Dictionary 3. As depicted in Figure 4, the reduction process disposes of Dictionary 2 and points Data Vertex B at Dictionary 1 (no change to the compressed data values is required as these dictionaries are identical). The subset Dictionary 4 is then removed with Data Vertex D amended to point to Dictionary 3. In this case the compressed data tokens stored in the data vertex are updated to reflect the encoding used by the new dictionary. The problem here is to compare each dictionary using the fewest possible file accesses. In the worst case each file would have to be checked against every other file to test for duplication ($n(n-1)/2$ comparisons) and for the existence of a subset (a further $n(n-1)$ comparisons). Comparisons can be reduced by using advance knowledge of dictionary file and token sizes gathered as the dictionaries are created. This means only dictionaries of the same token size are compared while file sizes can be used to determine whether to test for

Table 3. Query example

(a)	(b)
Uncompressed approach	Compressed approach
Note all A entries	Note all A vertices
Note all B entries	Search children of A to find B vertices
Note all entries matching "Data"	Search children of B to find "Data" entries
Retain B entries with links to "Data" entries	Retain B entries with links to "Data" entries
Retain A entries with links to remaining B entries	Retain A entries with links to remaining B entries
Return results	Return results

duplication or a subset and, in the case of the latter, which dictionary to treat as the potential superset and which to treat as the potential subset.

5 Results and Discussion

Figure 5 displays the results produced by the various bisimilarity strategies applied to the sample data sets. For each data set processed using the no bisimilarity option, only one data vertex is produced. With no bisimilarity, datagraph nodes are grouped by their label and just as all nodes of type `StaffMember` or `Student` would be grouped together, so too the `CDATA` nodes (which contain all the data values) are grouped into a single data vertex. The addition of forwards bisimilarity causes no change in the number of data vertices - each data set again having only one. Although forwards bisimilarity may have effects on the partitioning of structural nodes elsewhere in the structure, a method that exploits outgoing paths predictably has no effect upon the data-containing leaf nodes, as these have no descendant nodes for forwards bisimilarity to examine.

Working in the opposite direction, the use of full backwards bisimilarity results in an increase in the number of data vertices produced. By looking back up the datagraph at the ancestor nodes, the names of the XML entities in each data set are taken into account during partitioning and this leads to the single data vertex being split into multiple data vertices. Data set variation in the number of data vertices produced using backwards bisimilarity results from both the number of different XML entity names within each data set and the number of levels contained within each datagraph. The net result of partitioning using this method is that one data vertex is produced for each uniquely-named path through the datagraph. Full forwards and backwards bisimilarity alternately applies the bisimilarity rules in each direction until a stable structure is found. This means that a split caused by forwards bisimilarity higher up the datagraph can have an effect on the data vertices produced at the bottom of the structure when backwards bisimilarity considers the ancestors of each data node. The full forwards and backwards bisimilarity partitioning method is clearly influenced by the semi-structured nature of the test data sets. This is most apparent for XMark-30, where the irregular structure of the data set leads to a large number of data vertices when forwards and backwards bisimilarity is employed. This

Table 4. Path queries evaluated in Figure 8

Q#	Data Set	Query
Q1	XMark10	/regions/africa/item
Q2	XMark10	/regions/*/item
Q3	Legal-1	/sis/pc_age="21"
Q4	Legal-1	/sis/[pc_judge="J1" & pc_category="C1"]
Q5	Orders-1	/T/O_CUSTKEY="370"
Q6	Orders-1	/T/[O_ORDERSTATUS="O" & O_ORDER-PRIORITY="1-URGENT"]

form of bisimilarity produces the highest number of data vertices across all data sets with the exception of Orders-15 and ModifiedOrders-15. These data sets have a simple, regular structure which is unaffected by forwards bisimilarity - there are no variations in outgoing paths for any type of node.

The compressed size is the size of the data dictionaries plus the size of the compressed data values (as tokenised using the dictionaries). Figure 6 shows that as with results for the number of data vertices produced, the compression levels achieved for the data sets are the same for both the no bisimilarity and the full forwards bisimilarity partitioning methods. Again this is expected as the result of the partitioning process is the same for each method.

The distribution of the data values across the data vertices affects the impact of backwards bisimilarity. In the case of XMark-30 the introduction of backwards bisimilarity increases the compressed size over that produced using no bisimilarity. In this case the redistribution of the data values into a greater number of data vertices has led to a reduction in the overall levels of repetition within those data vertices consequently the dictionary-based scheme cannot operate as effectively and the compressed size rises.

The combined use of full forwards and backwards bisimilarity leads to an increase in the number of data vertices. This affects data value distribution and consequently the compressed data size. The increase in compressed size for XMark-30, Legal-1, Legal-13, Medline and NASA is caused by a reduction in the repetition of data values within the data vertices, while Dream, Rat and Human all reduce in compressed size as the increased distribution of data values across the data vertices separates the values in such as way as to allow the use of smaller token sizes. As previously noted, forwards bisimilarity has no effect upon the number of data vertices in the Orders-15 and ModifiedOrders-15 data sets and consequently has no effect on compressed sizes for these data sets either.

On balance it appears that, in terms of compressed data size, the full backwards bisimilarity partitioning method offers the greatest benefit for the majority of data sets. The significant exception to this is XMark-30 which experiences the best compression when using no bisimilarity (grouping by label only), and to a lesser extent the Dream, Rat and Human data sets which all slightly favour the full forwards and backwards bisimilarity method. However, the overall compressed size is only one factor when selecting a partitioning method, the number of data vertices produced must also be considered as this has an effect on individual data vertex size. Therefore, despite the slight adverse effect on the overall compressed size of some data sets, it is considered that the greater number of

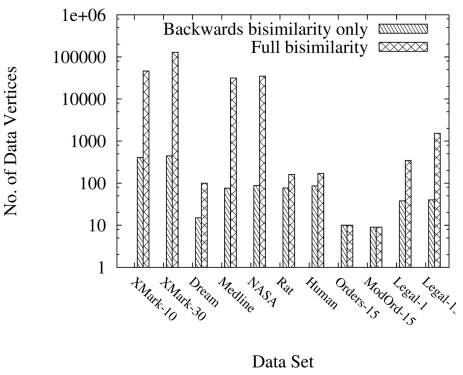


Fig. 5. Effect of bisimilarity on number of data vertices

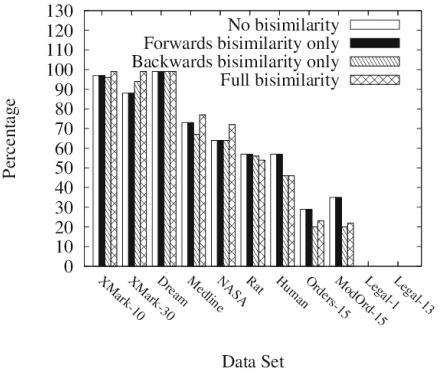


Fig. 6. Effects of bisimilarity on compressed sizes

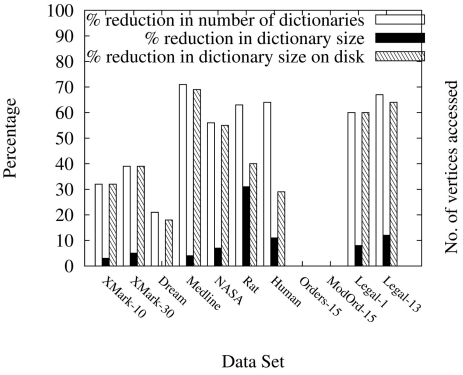


Fig. 7. Summary of dictionary reduction effects

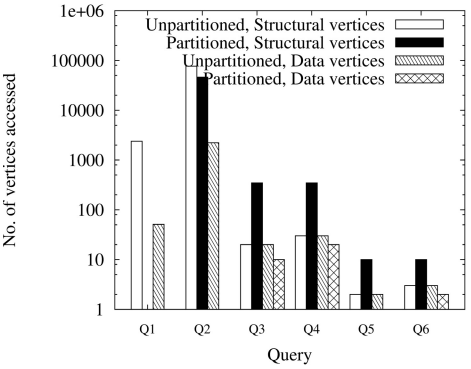


Fig. 8. Effect of partitioning the structure on query performance

data vertices produced by the full forwards and backwards bisimilarity method makes it the most reasonable compromise.

The work on querying the compressed structure, had two key objectives: first to demonstrate that the data values remained accessible in their compressed form and second to show that, in taking the partitioned structure into account, a query strategy could be formed that reduced the segments of the data structure required to be accessed to answer a query.

Query processing was evaluated by comparing the behaviour of the partitioned datagraph with the unpartitioned version. Dictionary structures were generated by using full F&B partitioning (Figure 8). Table 4 lists six queries performed over three of the test data sets - Legal-1, Orders-1 and XMark10. Performance is assessed by recording the number of vertices that need to be accessed to resolve each of the queries.

For structural queries (Q1 and Q2), the number of vertices accessed is reduced because the partitioned strategy accesses only those vertices which appear below the target node. As a purely structural query, neither query method accesses any data vertices. In Q2, the unpartitioned approach accesses all vertices, both structural and data, as it cannot distinguish between data and structural vertices in the context of the wildcard character.

For Q3 to Q6, the number of structural vertices required by each strategy does not differ between the query strategies. Given the short, regular structures of these data sets this is to be expected. There is however, variation in the number of data vertices. Due to the way in which the unpartitioned query strategy processes the query, each data vertex in the structure is accessed to check for matching data values. By contrast, the partitioned strategy only accesses the data vertices that satisfy the structural part of the query, reducing the pool of potential matches at each stage, so that a much smaller number of data vertices are required. In the case of Q3, the structure-minded strategy is able to narrow its search to only the 10 data vertices which hold `pc_age` values, as opposed to the full set of 346 data vertices that the unpartitioned strategy must access.

Changes to the total dictionary file sizes for each data set as a consequence of dictionary reduction are shown in Figure 7. For the Medline data set, 71% of its dictionaries have been removed by the reduction process, yet this leads to only a 4% reduction in logical dictionary size. This is because the dictionaries that have been removed are small. The large gap between the logical size reduction and the "size on disk" reduction stems from the removed dictionaries being much smaller than the 4Kb unit of disk space allocated by the filing system (the system was implemented on NTFS). Although this applies to every dictionary, as each logical file size is rounded up to the next 4Kb disk unit, the amount of wasted disk space depends on how close the dictionary is to filling the last disk unit allocated to that file. The effect is most pronounced on the smallest dictionaries, where the wasted disk space can form a much higher percentage of the "size on disk" allocated to the dictionary. It is also noted that for those dictionaries under a logical size of 2Kb, the wasted disk space will exceed that logically required by the dictionary entries. Where the logical sizes of the dictionaries removed are larger, such as with the Rat data set, the gap between logical size and "size on disk" is much lower as the wasted disk space forms a smaller percentage of the disk space allocated by the filing system.

A benefit of the dictionary-based scheme is that only those dictionaries actually involved with a particular query need to be accessed by the query system. This has implications where files are being requested over a data connection. In such a scenario the bandwidth utilisation would be limited to only those dictionaries that are useful. This is in contrast to other queryable ([8], [9], [10], [11]) for which the data structure must be transferred as a complete single entity.

It is not simply the total size of the dictionaries that is a potential benefit of dictionary reduction. While these savings in storage space are useful, the removal of the redundant dictionaries also means that the surviving dictionaries may relate to more than one data vertex within the datagraph. This increases

the likelihood that additional user queries may be satisfied using dictionaries that have already been acquired.

Further improvements are possible by combining dictionaries with overlapping contents where the two dictionaries use the same size of token. Such a method would reduce the number of dictionaries without impacting on the compressed data size. However, any method of combining dictionaries that causes the token sizes within the compressed data to increase must be treated with caution, as the reduction in terms of overall dictionary size could easily be negated by the consequent increase in compressed data size. A trade-off is possible between dictionary numbers and compressed data sizes that allows for less wastage of the allocated storage space. In some cases, tokens can be represented in fewer bits than are used to represent an integer. In addition, further compression (eg Huffmann coding) may usefully be applied to the dictionaries. This could be particularly helpful where large text elements are contained within the data values, such as those in the Orders data set, as these are currently stored in the dictionaries in their original uncompressed form. While this could have benefits in terms of space occupied by the dictionaries, there is likely to be some impact on query performance.

6 Conclusion

The aim of this work has been the evaluation of a data storage model that facilitates the sharing of individual segments of data while maintaining support for user queries. It was proposed that this could be achieved using a combination of bisimilarity-based partitioning and dictionary-based compression. Work on the integration of data value compression described the steps necessary to build dictionary-based compression into datagraphs. The construction of dictionaries and the subsequent encoding of data values provides the basis for a query strategy that can deal with compressed values and take advantage of the partitioned structure. It was demonstrated that data values remained queryable in their compressed form and that the use of a structure-aware query strategy enabled the evaluation of queries using only the relevant parts of the data structure.

The contribution of this work is the evaluation of a data storage model that combines bisimilarity-based partitioning and dictionary compression methods. The evidence presented suggests that this approach has benefits in terms of data storage. Support for queries is not only maintained but also demonstrated to access only a fraction of the entire data set. The resulting structure is such that it lends itself to future exploitation in a system that shares independent segments of data.

References

1. Liefke, H., Suciu, D.: XMILL: An Efficient Compressor for XML Data. In: Proc ACM SIGMOD, pp. 153–164 (2000)
2. Cheney, J.: Compressing XML with Multiplexed Hierarchical PPM Models. In: Data Compression Conference (DCC 2001), pp. 163–172. IEEE Computer Society (2001)

3. Levene, M., Wood, P.: XML Structure Compression. In: International Workshop on Web Dynamics (2002)
4. Skibinski, P., Grabowski, S., Swacha, J.: Fast Transform for Effective XML Compression. In: Proc CADSM, pp. 323–326 (2007)
5. Tolani, P.M., Haritsa, J.R.: XGRIND: A Query-Friendly XML Compressor. In: ICDE 2002, pp. 225–234 (2002)
6. Min, J.K., Park, M.J., Chung, C.W.: XPRESS: A Queriable Compression for XML Data. In: Proc ACM SIGMOD, pp. 122–133. ACM (2003)
7. Skibiński, P., Swacha, J.: Combining efficient XML compression with query processing. In: Ioannidis, Y., Novikov, B., Rachev, B. (eds.) ADBIS 2007. LNCS, vol. 4690, pp. 330–342. Springer, Heidelberg (2007)
8. Arion, A., Bonifati, A., Costa, G., D'Aguanno, S., Manolescu, I., Pugliese, A.: Efficient query evaluation over compressed XML data. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 200–218. Springer, Heidelberg (2004)
9. Cheng, J., Ng, W.: XQzip: Querying compressed XML using structural indexing. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 219–236. Springer, Heidelberg (2004)
10. Ng, W., Lam, W.Y., Wood, P.T., Levene, M.: XCQ: A queriable XML compression system. Knowledge and Information Systems 10(4), 421–452 (2006)
11. Wong, R.K., Lam, F., Shui, W.M.: Querying and maintaining a compact XML storage. In: Proc WWW Conference, pp. 1073–1082. ACM (2007)
12. Kaushik, R., Shenoy, P., Bohannon, P., Gudes, E.: Exploiting Local Similarity for Indexing Paths in Graph-Structured Data. In: Proc ICDE 2002, pp. 129–140 (2002)
13. Buneman, P., Grohe, M., Koch, C.: Path Queries on Compressed XML. In: Proc 29th VLDB, pp. 141–152 (2003)
14. Schroeder, R., Mello, R., Hara, C.: Affinity-based xml fragmentation. In: 15th WebDB (2012)
15. Alghamdi, N., Rahayu, W., Pardede, E.: Object-based methodology for xml data partitioning (oxdp). In: Proc IEEE AINA, pp. 307–315. IEEE (2011)
16. Marian, A., Siméon, J.: Projecting xml documents. In: Proc 29th VLDB, pp. 213–224. VLDB Endowment (2003)
17. Bidoit, N., Colazzo, D., Malla, N., Sartiani, C.: Partitioning xml documents for iterative queries. In: Proc 16th IDEAS, pp. 51–60. ACM (2012)
18. Goldman, R., Widom, J.: DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In: VLDB 1997, pp. 436–445 (1997)
19. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann (1999)
20. Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F.: Covering Indexes for Branching Path Queries. In: Proc ACM SIGMOD, pp. 133–144. ACM (2002)
21. Wilson, J., Gourlay, R., Japp, R., Neumuller, M.: Extracting partition statistics from semistructured data. In: Proc 16th DEXA, pp. 497–501. IEEE (2006)
22. Dietz, P.F.: Maintaining Order in a Linked List. In: Proc 14th ACM STOC, pp. 122–127. ACM (1982)
23. Tripney, B., Foley, C., Gourlay, R., Wilson, J.N.: Sharing large data collections between mobile peers. In: Proc 7th MoMM, pp. 321–325. ACM (2009)

Implementing Efficient Updates in Compressed Big Text Databases

Stefan Böttcher, Alexander Bültmann, Rita Hartel, and Jonathan Schlüßler

University of Paderborn, Computer Science, Fürstenallee 11, 33102 Paderborn, Germany
{stb,alexb,rst}@upb.de, sjonny@mail.upb.de

Abstract. Text compression techniques like bzip2 lack the possibility to insert or to delete strings at a given position into a text that has been compressed without prior decompression of the compressed text. We present a technique called DICIRT that supports fast insertion into and deletion from compressed texts without full decompression of the compressed text. For inserted fragments up to a size of 8% of the original text size, and for deleted fragments up to 15% of the original text DICIRT is faster than modifying uncompressed text preceded by a decompression step and followed by a compression step.

Keywords: compression, BWT, wavelet trees, modification of compressed text.

1 Introduction

1.1 Motivation

Applications like big data collections, column-oriented databases, main memory databases, XML compression, and data exchange with mobile client devices benefit from using compressed texts in terms of space consumption, data access time, and data transfer time. When all the text values of a database column or of the leaf nodes in an XML document are regarded as the words of a text S and are compressed together, typical operations like reading or updating the n^{th} row of a database column or the n^{th} leaf node of an XML document, requires reading or updating the n^{th} word of compressed texts. When such a text S, e.g. a database column S or a sequence S of XML leaf nodes, is stored as bzip2 compressed format c(S) only, the modification of the n^{th} word of S, e.g. the n^{th} row in a database column or the n^{th} leaf node in an XML document could be performed by decompressing c(S) to S, modifying S to S', compressing S' back to c(S'), and storing c(S'). However, if time and space limitations are the bottleneck of an application that requires inserting, deleting, or updating words at a given text position, say, the n^{th} word of a huge text S, it can be a significant advantage to perform these operations directly on the compressed text c(S), in comparison to decompressing c(S) to S, doing the insert, update, or delete operation on S, and eventually compress a modified version S' of S to c(S').

Our contribution is based on a block-sorting technique called Indexed Reversible Transformation (IRT) [1] that modifies Burrows Wheeler Transformation (BWT) [2]

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 189–202, 2013.

in such a way that search, insertion, and deletion of the n^{th} word are supported on the block-sorted text IRT(S). IRT(S) can then be compressed to CIRT (=compressed IRT) with run length encoding (RLE) and wavelet trees (WT). Beyond our previous contribution [1], we present a technique called DICIRT (=Deletion and Insertion in CIRT) to directly insert the n^{th} word into or to directly delete the n^{th} word from a compressed text c(S) without full decompression of c(S), where c(S) is computed from S by CIRT.

As our evaluation shows, insertions into a CIRT compressed text are faster than decompression, insertion, and compression using bzip2 for inserted fragments up to a size of 8% of the uncompressed text. Furthermore, deletions from a CIRT compressed text are faster than decompression, deletion, and compression using bzip2 for deleted fragments up to a size of even 15% of the uncompressed text.

Furthermore, we have combined DICIRT with XML compression techniques like DAG compression [3], Succinct [4], and Repair [5], each of which allows updates on compressed XML structures and can identify the node IDs of XML text nodes that have to be updated as a consequence of an update operation on a compressed XML tree. Therefore, for each of these XML compression techniques, using CIRT as compressed text format and using DICIRT as update processor for XML text nodes is considered to be superior to text compression with bzip2 for text inserts up to 8% and text deletions up to 15% of the total size of all text nodes.

1.2 The Advantage of Using IRT in Comparison to Using BWT

The major advantage of IRT in comparison to BWT is that insertion and deletion of words commute with the transformation and retransformation of a text when using IRT, which is not the case for BWT (c.f. Figure 1). For example, let BWT(S-W) denote BWT applied to a text S-W, where S-W is a text S from which a word W has been deleted, and let BWT(S)-BWT(W) denote BWT(S) from which those characters have been deleted that represent the word W in BWT(S). Let IRT(S)-IRT(W) and IRT(S-W) be defined correspondingly, except using IRT instead of BWT. Then, BWT(S-W) is not guaranteed to be equal to BWT(S)-BWT(W), i.e., BWT transformation and word deletion do not commute (Figure 1b). However, IRT transformation commutes with both, insertion and deletion of words, such that insertion and deletion of words can be done directly on IRT(S) (Figure 1a). This includes that even inserting IRT(W), i.e. the transformed word W, at the word position n into IRT(S-W) returns the same result as IRT applied to the text S resulting from W being inserted as the n^{th} word into S-W.

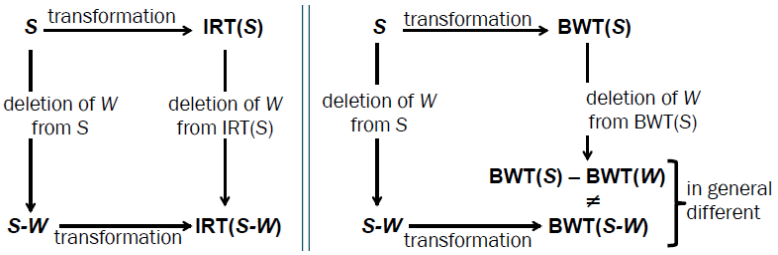


Fig. 1. (a) IRT commutes with insert and delete of words, while BWT does not (b)

1.3 Related Work

Previous contributions to the field of text compression can be classified into entropy encoders, e.g. [6], [7], [8], arithmetic coders, e.g. [9], [10], dictionary coders, e.g. [11], [12], [13], statistical compressors, e.g. [14], [15], grammar-based compressors, e.g. [16], and block-sorting compressors. While previous contributions based on dictionary coders, e.g. [17], or based on grammar-based compressors, e.g. [18], allow random access to arbitrary phrases of the compressed text, common block-sorting compressors like bzip2 that are based on Burrows-Wheeler-Transformation (BWT) combined with Move-To-Front and Huffman Encoding [6] usually do not allow for direct access to arbitrary parts of compressed data. Therefore, searching, modifying and partial decompression are not possible without an additional index on text compressed with these compressors, but are features of our approach DICIRT which is based on a variation of BWT.

Other contributions based on BWT, e.g. [19] and [20], propose efficient pattern matching in text libraries or pattern collections and provide a possibility to add or to remove new texts or patterns. However, patterns cannot be associated with a position, e.g. containing the n^{th} word. Therefore, text collections as described in [19], [20] and compressors allowing random access to the compressed text as described in [17], [18], [21] and [22] cannot be efficiently used for position-based insertion, deletion or modification of the n^{th} word in the compressed text without prior decompression. As a consequence, these approaches cannot be used to insert, to delete, or to update the n^{th} text node in a compressed XML file.

Further contributions propose update algorithms for BWT, which allow for insertions, deletions and substitution of letters without retransforming BWT, e.g. [23] and [24]. But equally to [17], [18], [19], and [20], they do not allow for position-based insertion, deletion or modification of the n^{th} word within transformed texts, i.e., they do not provide a solution for insertion, deletion or modification of the n^{th} text node of a compressed XML file. Furthermore, position-based word search of the n^{th} word in compressed texts, if possible at all, requires usually an external index, when using previous text compressors.

In comparison to these approaches, we follow the idea of Indexed Reversible Transformation (IRT) [1], a self-index to block-sorted texts, which allows for position-based search, insert, update, and delete operations, e.g. of the n^{th} word of a text, without full decompression of the text. While IRT(S) can be compressed to CIRT(S) when using run-length FM-index (RLFMI) [25] and wavelet trees (WT) [26], CIRT(S) loses some of IRT's benefits: Either insert, update, and delete operations on RLFMI and WT take much time or decompression of RLFMI and WT back to IRT(S) is required before the insertion or deletion on IRT(S), as well as a compression back to CIRT(S) afterwards.

In this paper, we present DICIRT, a technique that supports insert and delete operations directly on the compressed data structure, i.e. on RLFMI and on WT, without full decompression of RLFMI and WT. As a consequence, faster updates of the compressed block-sorted texts are possible, and, as a direct consequence, faster updates of text nodes in compressed XML are possible.

1.4 Contributions

This paper proposes a technique called DICIRT that provides fast insert, update and delete operations on texts compressed by the *Compressed Indexed Reversible Transformation* (CIRT) [1], which already offers the following properties:

- searching for positions of words in S, given the word's content and CIRT(S),
- searching in the compressed text CIRT(S) for the n^{th} word of S,
- retransforming the n^{th} word of S from CIRT(S), and
- searching for words being $\leq, <, \geq, >, =$ or $\neq$ a given text constant.

Beyond [1], DICIRT supports the following operations directly on the compressed text CIRT(S) without prior decompression of CIRT(S) to IRT(S) or to S and without time consuming bit-shift operations on compressed data:

- fast deletion of the n^{th} word of S from the compressed text CIRT(S),
- fast insertion of a word as the n^{th} word of S into the compressed text CIRT(S), and
- fast merging of another transformed and compressed text CIRT(W) into CIRT(S) simulating an insertion of W behind the n^{th} word of S.

We furthermore present a comprehensive evaluation, which shows that DICIRT is faster for deletions and insertions than compressing and decompression text with bzip2 if the deleted or inserted text fragments are small compared to the given text.

2 Fast Insertion and Deletion on CIRT

2.1 Indexed Reversible Transformation (IRT) and Used Notation

Let S be the concatenation of words, such that each word starts with a word delimiter (e.g. '$') and continues with a possibly empty sequence of characters not containing the word delimiter. For the n^{th} word of S, we call n the word index of the word in S. Furthermore, let $|S|$ denote the length of S, $|S|_W$ denote the number of words in S, $S[n]$ denote the n^{th} character of S, and $S[n..n+i]$ denote the sequence of characters ranging from $S[n]$ to $S[n+i]$ for $i \geq 0$.

The Indexed Reversible Transformation (IRT) [1] of S denotes the Burrows-Wheeler-Transformation (BWT) [2] applied to S according to an ordering relation $A_\$$ that fulfills the following conditions. The lexicographical order of the word delimiters '$' is changed in such a way, that all '$' of S get the smallest lexicographical order in $A_\$$, and most important, the order of the word delimiters among themselves is determined by their occurrence in S from left to right. That is, the n^{th} word delimiter '$' appearing in S gets a smaller lexicographical order in $A_\$$ than the $n+1^{st}$ word delimiter '$'. Furthermore, IRT sorts characters of S according to their prefix (opposed to BWT that sorts them according to their suffix). Thus, in contrast to BWT, for each $n \in \{1,\ldots,|S|_W\}$, IRT(S) stores the first character of the n^{th} word of S at position n of IRT(S). This provides an index to the first character of each word of S, which allows for reconstruction of each word individually without retransforming IRT(S) in total.

Let IRT(S) denote IRT applied to the text S. Let Sort(S) denote S and IRT(S) lexicographically sorted according to a given ordering relation $A_\$$. Let B(IRT(S)) denote the run-length bit-vector of IRT(S), defined in RLFMI [25] and shown in Figure 2, that contains a 0-bit for each character in IRT(S)) that is equal to the previous character and a 1-bit otherwise. Let R(IRT(S)) denote IRT(S) after replacing each run of equal characters within IRT(S) by one character. Finally, let WT(R(IRT(S))) be the wavelet tree containing the bits of the Huffman codes of all characters c_i of R(IRT(S)) as described in [26] and shown e.g. in the left half of Figure 3, such that for each c_i,

(1) the Huffman code of c_i is presented in the wavelet tree as a path from the root node to the leaf node representing c_i in WT,
(2) one bit of the Huffman code of c_i is stored on each node of this path except for the leaf node,
(3) if a 0-bit is stored in a node, the path continues with the left sub-tree, if a 1-bit is stored in a node, the path continues with the right sub-tree.

2.2 Delete Operation on CIRT(S)

The deletion of a word W on CIRT(S), i.e. on B(IRT(S)) and WT(R(IRT(S))), where CIRT(S') denotes the result of the deletion, is done in two steps. In a first step, B(IRT(S)) is updated. For this purpose, W is transformed consecutively from CIRT(S), i.e. B(IRT(S)) and WT(R(IRT(S))), and the visited bits of B(IRT(S)) are marked by using e.g. a bit vector of length $|S|$. Then, B(IRT(S')) is initialized as a bit-vector having length $|S|$-$|W|$, and the bits of B(IRT(S)), which are not marked and do not follow a marked bit, are copied bit by bit to B(IRT(S')). Whenever a marked bit in B(IRT(S)) at position p is reached, the value of the next unmarked bit at position p_n in B(IRT(S)) has to be verified by checking the equality of IRT(S)[p_p] and IRT(S)[p_n], where p_p is the position of the previous unmarked bit before p. If IRT(S)[p_p] = IRT(S)[p_n], a 0-bit has to be written to B(IRT(S'))[p_n], regardless of whether the bit value of B(IRT(S))[p_p] is 0 or 1. If IRT(S)[p_p] ≠ IRT(S)[p_n] or no previous unmarked bit exists, a 1-bit has to be written to B(IRT(S'))[p_n], as IRT(S)[p_p] and IRT(S)[p_n] will not be part of the same run in IRT(S').

The example in Figure 2 shows the normal copying of bits and both possible special cases by deleting the second word from S = "abraca$dabra". The bits at the unmarked positions of B(IRT(S)) are copied to B(IRT(S')). The bits of B(IRT(S)), which have to be further considered are marked by a question mark. For example the characters 'b' at position 4 and 'b' at position 6 in IRT(S) form a run after the deletion of the character '$' at position 5, i.e., instead of copying the 1-bit at position 6 in B(IRT(S)), a 0-bit has to be written to B(IRT(S')). On the other hand, the character at position 12, which does not start a run in IRT(S), starts a run in IRT(S') because of the deletion of the 'a' at position 11. Therefore, instead of the 0-bit at position 12 of B(IRT(S)), a 1-bit has to be written to B(IRT(S')). All bits that follow a marked position might change when transferring them from B(IRT(S) to B(IRT(S')), as runs might be joined here. Note that all other bits can be copied from B(IRT(S)) to B(IRT(S')) without further consideration.

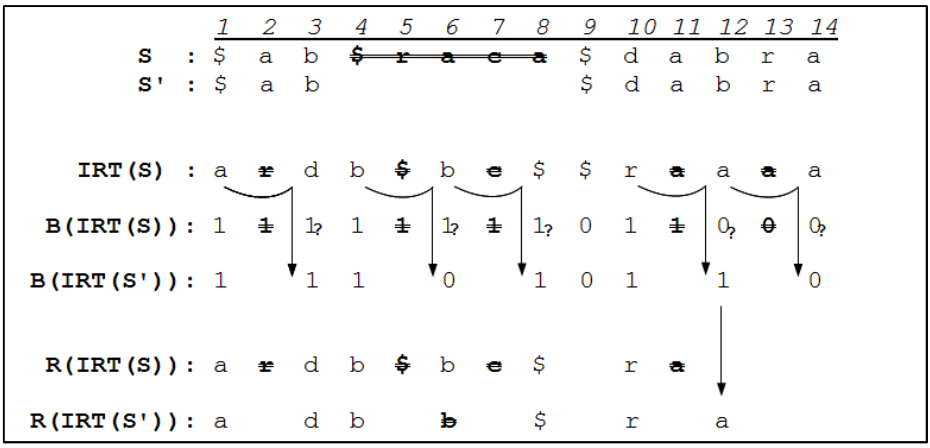


Fig. 2. Deletion of the 2nd word of S = "abraca$dabra"

In a second step, WT(R(IRT(S))) has to be updated according to the deletion of characters from R(IRT(S)). Note that the number of 1-bits in B(IRT(S')) cannot increase in comparison to the number of 1-bits in B(IRT(S)), i.e., no new runs can occur in IRT(S') and therefore, no new characters can occur in R(IRT(S')). A 0-bit in B(IRT(S)) can only change to a 1-bit, if the start-bit of the corresponding run is deleted (for example, see Figure 2, position 12), which does not change R(IRT(S')).

As R(IRT(S)) is stored as WT(R(IRT(S))), the characters to be deleted from R(IRT(S)) have to be deleted from WT(R(IRT(S)) to yield WT(R(IRT(S')). For this purpose, to delete the n^{th} character from WT(R(IRT(S))), in a first step, the n^{th} bit in the root of WT(R(IRT(S))) is marked while processing B(IRT(S)). Second, the marking of bits to be deleted is propagated down the WT, i.e., all bits which represent the n^{th} character in WT(R(IRT(S))) are marked. WT(R(IRT(S'))) is initialized as an empty data structure with equal shape to WT(R(IRT(S))). The size of each node in WT(R(IRT(S'))) is determined by the number of unmarked bits in the corresponding node in WT(R(IRT(S))). Finally, all unmarked bits in each node of WT(R(IRT(S))) are copied to the corresponding node in WT(R(IRT(S'))).

The example in Figure 3 shows WT(R(IRT(S))) and WT(R(IRT(S'))) for the example delete operation on S shown in Figure 2. The characters that shall be deleted from R(IRT(S)) and the bits that shall be deleted from the nodes of WT(R(IRT(S))) are crossed out in WT(R(IRT(S))) shown in Figure 3 and are not copied to WT(R(IRT(S'))). Note that the characters shown in the inner nodes of the wavelet tree in Figure 3 are not part of the wavelet tree, but are shown to illustrate the meaning of the bits in the wavelet tree.

2.3 Insert Operation on CIRT(S)

The insertion of a word W as the n^{th} word of S into CIRT(S), i.e. into B(IRT(S)) and WT(R(IRT(S))), starts by compressing W to CIRT(W), where WT(R(IRT(W))) is

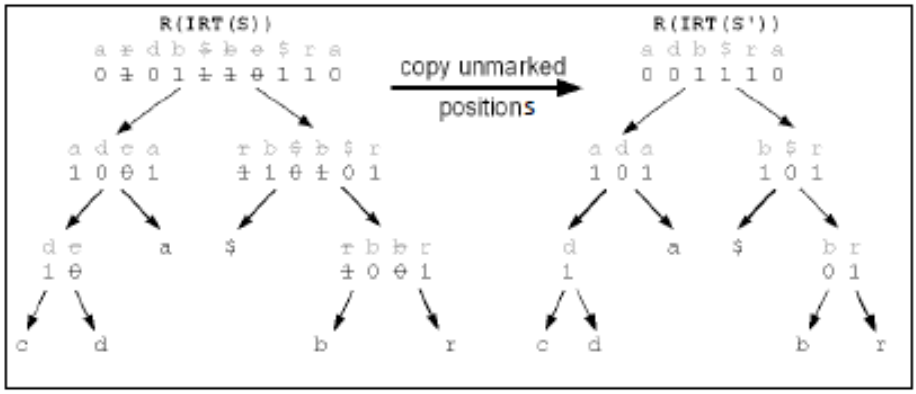


Fig. 3. R(IRT(S)) and R(IRT(*S*')) compressed as WT. The characters shown are not part of the WT, i.e., the WT only contains the bits of the characters' Huffman codes.

shaped like WT(R(IRT(*S*))). Let CIRT(*S*') denote the result of the insertion. Then, the insertion of CIRT(*W*) into CIRT(*S*) is done by initializing B(IRT(*S*')) and WT(R(IRT(*S*')) as empty data structures and by the following three steps.

The first step is the Insert Position Computation (IPC) Algorithm (below), which works similar to the merge operation described in [1]. IPC determines the positions Ipos, where the bits of B(IRT(*W*)) have to be inserted into B(IRT(*S*')). The n^{th} bit of B(IRT(*S*')) represents the first character W[1] of *W*, because *W* is inserted as the n^{th} word into CIRT(*S*). That is, the insert position for *W*[1] is n=pS+pW (line (4)), where pW denotes the number of characters from *W* up to the current insert position pS+pW and pS denotes the number of characters from *S* up to the current insert position pS+pW. To find the successor insert position representing *W*[i+1] in B(IRT(*S*')) of a current insert position pS+pW representing *W*[i] in B(IRT(*S*')), we have to recalculate pS and pW. The new value for pW calculates the position of W[i+1] in IRT(*W*) by LF mapping (lines (5)+(6)). And the new value for pS calculates how many characters from IRT(*S*) occur before the insert position of the new character (line (7)).

(1) Initialize insert positions Ipos[1..|*S*|+|*W*|] with 0-bits ;
(2) pW = 1 ; // pW = how many characters until n are from IRT(*W*)
(3) pS = n – 1 ; // pS = how many characters until n are from IRT(*S*)
(4) do { Ipos[pS+pW] = 1 ; // mark this insert position for actual char from IRT(*W*)
(5) char = CIRT(*W*)[pW] ; // get actual char from IRT(*W*)
(6) pW = count(*W*)[char] + $Rank_{char}$(CIRT(*W*),pW) ; // LF mapping on IRT(*W*)
(7) pS = count(*S*)[char] + $Rank_{char}$(CIRT(*S*),pS) ;
(8) } while (char ≠ '$') // until all chars of IRT(*W*) are processsed

IPC Algorithm: mark insert positions for bits from B(IRT(*W*)) in B(IRT(*S*'))

Note that pS (and pW) can be computed by using count and Rank only, where $Rank_c$(CIRT(*S*), *p*) denotes the number of characters c up to position *p* in IRT(*S*) computed using only B(IRT(*S*)) and WT(R(IRT(*S*))), and count(*S*)[char] denotes the number of characters in *S* that are alphabetically smaller tham char. Furthermore, the

assignment in line (5) returns the character at position pW in IRT(W) using only B(IRT(*S*)) and WT(R(IRT(*S*))). This is repeated for all characters of *W*[i] and in such a way that all the visited positions in B(IRT(*S*')) are marked by using the additional bit-vector Ipos (line (4)).

In a second step, for all positions *p*, the bits of B(IRT(*S*')) are copied from B(IRT(*S*)) and from B(IRT(*W*)) starting at the first position *p* of B(IRT(*S*')). If position *p* of B(IRT(*S*')) is marked, the next bit is copied from the next position of B(IRT(*W*)) to position *p* of B(IRT(*S*')), otherwise from the next position of B(IRT(*S*)). When merging IRT(*S*) and IRT(*W*), new runs in IRT(*S*') may occur, or runs copied from IRT(*S*) or IRT(*W*) may be joined. At every transition from a marked position *p*-1 to a non-marked position *p* or vice versa from a non-marked position *p*-1 to a marked position *p*, the value of the copied bit has to be verified according to its character value in IRT(*S*) resp. IRT(*W*) similar to section 2.2. If the character at position *p* in IRT(*S*') is preceded by an equal character at position *p*-1 in IRT(*S*'), a 0-bit is written to position *p* of B(IRT(*S*')), otherwise a 1-bit.

The example in Figure 4 shows the insertion of the word *W* = „$raca“ into *S* = “$ab$dabra” as the 2nd word simulated on B(IRT(*W*)) and B(IRT(*S*)). *S*'=“abraca$dabra” denotes the result of the insertion of *W* into *S*. Within the first step, the insert positions in B(IRT(*S*')) are computed. They are represented by the character '_' within $[IRT(S')]_S$ which itself denotes IRT(*S*') projected to the elements of S. Similarly, $[IRT(S')]_W$ denotes IRT(*S*') projected to the elements of *W*. Furthermore, $\{B(IRT(S))\}_{S'}$ denotes the mapping of the bits of B(IRT(*S*)) to the positions of $[IRT(S')]_S$, and $\{B(IRT(W))\}_{S'}$ denotes the mapping of the bits of B(IRT(*W*)) to the positions of $[IRT(S')]_W$. If the bits at the positions *p* and *p*-1 in B(IRT(*S'*)) are from the same source, either B(IRT(*S*)) or B(IRT(*W*)), the bit at position *p* can be copied from the source to B(IRT(*S'*)) without verification. Verification of bits in B(IRT(*S*')) is only necessary for those bits, for which the left neighbor bit in B(IRT(*S*')) has been copied from the other source, B(IRT(*W*)) or B(IRT(*S*)) respectively. For example, instead of the 0-bit of B(IRT(*S*)), a 1-bit has to be written to position 6 of B(IRT(*S*')), as the 'b'-run of IRT(*S*) is interrupted by the '$' of IRT(*W*) at position 5 of B(IRT(*S*')). At position 12 of B(IRT(*S*')), instead of the 1-bit of B(IRT(*S*)), a 0-bit has to be written to B(IRT(*S*')), as the 'a' of IRT(*S*), is part of an 'a'- run leaded by a previous 'a' in IRT(*S*').

In a third step, which can be performed in parallel to the second step, the resulting wavelet tree WT(R(IRT(*S*'))) is constructed. For this purpose, the nodes of WT(R(IRT(*S*'))) are initialized as an empty data structure with an estimated size depending on the sizes of the corresponding nodes in WT(R(IRT(*W*))) and in WT(R(IRT(*S*))). Note that WT(R(IRT(*W*))) and WT(R(IRT(*S*))) must have the same shape, i.e. use the same Huffman coding, to perform the insert operation. Whenever a 1-bit is inserted at position *p* into B(IRT(*S*')), a character c has to be inserted into WT(R(IRT(*S*'))) at position $p_{WT} = Rank_1(B(IRT(S'), p)$. The character c is either copied from WT(R(IRT(*W*))) or from WT(R(IRT(*S*))), depending on whether the 1-bit has been copied from B(IRT(*W*)) or from B(IRT(*S*)) to B(IRT(*S*')). If a 0-bit of B(IRT(*W*)) or of B(IRT(*S*)) has been changed to a 1-bit when writing it to B(IRT(*S*')) (see for example the bit at position 6 in Figure 4), then depending on where the

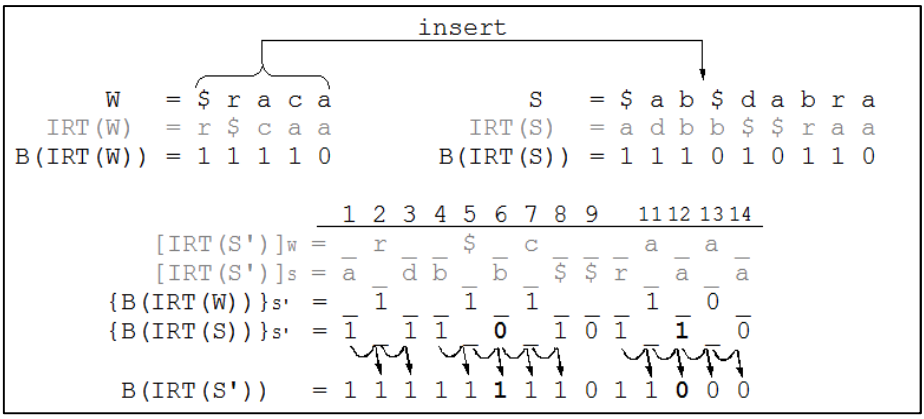


Fig. 4. Insertion of W = "$raca" as 2nd word into S = "abdabra" simulated on B(IRT(S)) and B(IRT(W))

changed 0-bit comes from, B(IRT(W)) or B(IRT(S)), the last copied character from B(IRT(W)) or from B(IRT(S)) respectively has to be copied to WT(R(IRT(S'))) again.

The insertion of the characters of W into S can be simulated on $[\text{IRT}(S')]_W$ and on $[\text{IRT}(S')]_S$ as illustrated in Figure 4, however it is implemented on WT(R(IRT(W))) and WT(R(IRT(S))) as follows (c.f. Figure 5). For each 1-bit in B(IRT(S')), a character from either WT(R(IRT(W))) or WT(R(IRT(S))) is copied to WT(R(IRT(S'))). For example, for the 1-bit at position 4 in B(IRT(S')) and for the 1-bit at position 6 in B(IRT(S')), the character 'b' occurring at position 3 in WT(R(IRT(S))) shown in Figure 5 is copied to WT(R(IRT(S'))). Note that the character 'b' occurring at position 3 in WT(R(IRT(S))) is copied twice to WT(R(IRT(S'))), i.e. to the index positions 4 and 6, because the original 'b'-run in IRT(S) shown in Figure 4 is split into 2 'b'-runs in IRT(S') because of the insertion of the '$'-character from IRT($W$) at position 5 into IRT(S') (c.f. Figure 4, positions 4-6 of $[\text{IRT}(S')]_W$ and $[\text{IRT}(S')]_S$).

A character at position q in WT(R(IRT(S))) or in WT(R(IRT(W))) is copied to WT(R(IRT(S'))) as follows. The character at position q is decompressed, and the bits visited in the wavelet tree nodes of WT(R(IRT(S))) or of WT(R(IRT(W))) respectively are added to their corresponding nodes in WT(R(IRT(S'))), in the same order as the characters represented by WT(R(IRT(S))) and by WT(R(IRT(W))) occur in the sequence of character represented by WT(R(IRT(S'))).

Finally, as the sizes of the nodes of WT(R(IRT(S'))) were estimated, the space required for a node might have to be adjusted if it is exceeded or if some space is unused after the insertion is completed.

If the insertion steps 2 and 3 are performed in parallel, the insertion of a character into WT(R(IRT(W))) can be used for improving the character comparison in the second step. The check of whether or not two characters c_1 and c_2 are equal requires decompression of c_1 and c_2 from their wavelet trees. One decompression step can be saved by saving the bits visited while decompressing a character of the input WT and by using the bits for insertion into the result WT if one of the characters to be compared is represented by a 1-bit in the B-vector of RLFMI.

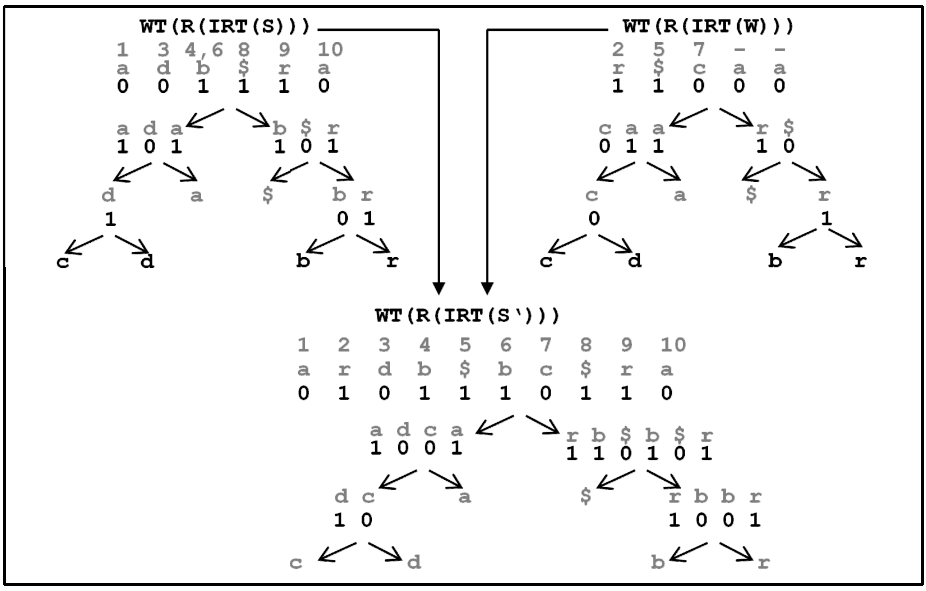


Fig. 5. Insertion of *W* as the 2nd word into *S* implemented on WT(R(IRT(*W*))) and WT(R(IRT(*S*)))

3 Evaluation

We have implemented DICIRT and compared it with bzip2 and CIRT, w.r.t. the compression ratio and the performance. The used datasets for the comparison of the compression ratio are:

Name	Description	Size
Bible	The Bible (in German)	4,323,185 bytes
Kant	"Kritik der reinen Vernunft" written by Immanuel Kant (in German)	1,302,886 bytes
Robinson	"The Life and Adventures of Robinson Crusoe" (in English)	625,504 bytes

DICIRT has a different compression ratio than CIRT [1] because it relies on wavelet-trees with a fixed shape. The wavelet-tree shape that we used in these tests was determined by the character frequency of the "Bible" dataset, thus yielding almost the same compression ratio for this dataset for CIRT and DICIRT (c.f. Figure 6). In comparison to CIRT, the compression ratio of DICIRT is slightly worse for the "Kant" and "Robinson" datasets. The compressor bzip2 compresses the datasets with about 63% of the size needed by CIRT.

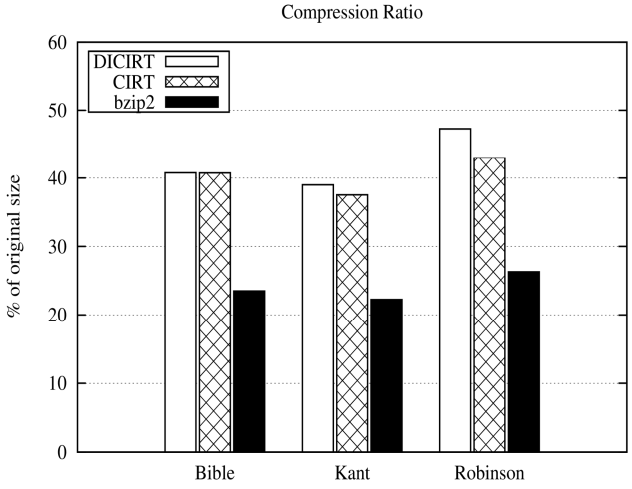


Fig. 6. Compression ratios of DICIRT, CIRT, and bzip2 on different datasets

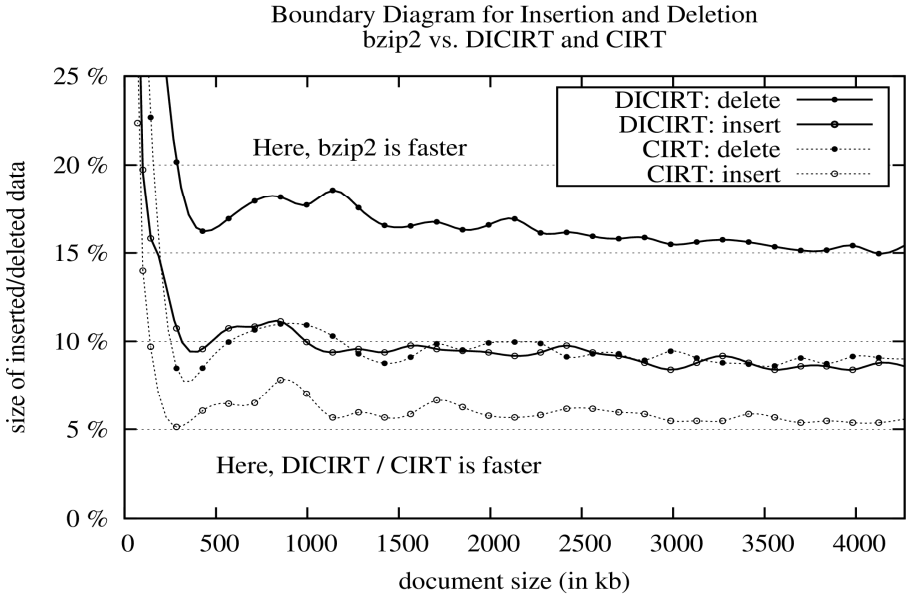


Fig. 7. Insert and delete boundary for DICIRT and CIRT vs. bzip2

The performance was evaluated by inserting data into and by deleting data from the datasets. When using CIRT [1], the compressed data first has to be retransformed to IRT(S), followed by the operation on IRT(S) resulting in IRT(S'), and a retransformation to CIRT(S'). bzip2 requires a complete decompression to the original string S, performing the operation on S, and compressing the result again. With DICIRT, the insert and delete operations can be performed directly on CIRT(S).

The performance of DICIRT is best for small insertions and deletions. In comparison to bzip2, DICIRT is up to 40% faster for single word operations. When it comes to operations where the deleted or inserted fragment is big, DICIRT is slower than bzip2. The boundary up to which DICIRT is superior to bzip2 was examined by using different parts of the "Bible" dataset and performing the insert and delete operation on each of them (c.f. Figure 7). This reveals that the insert boundary is at 8% and the delete boundary is at 15%, i.e., insertion of a fragment that has a size of less than 8% of the given text is faster when using DICIRT than when using bzip2, and the deletion of a fragment that has a size of less than 15% of the given text is faster when using DICIRT than when using bzip2.

Furthermore, we have compared the performance of DICIRT and its predecessor CIRT for insertions into and deletions from the "Bible" document. Figure 8 summarizes the results for insertions and deletions up to a size of 15% of the original document. Notice that for both, insertions and deletions, DICIRT clearly outperforms insertions and deletions on CIRT as described in [1], i.e., it is an advantage to do the deletions and insertions directly on B(IRT(*S*)) and WT(R(IRT(*S*))), in comparison to decompressing B(IRT(*S*)) and WT(R(IRT(*S*))) back to IRT(*S*).

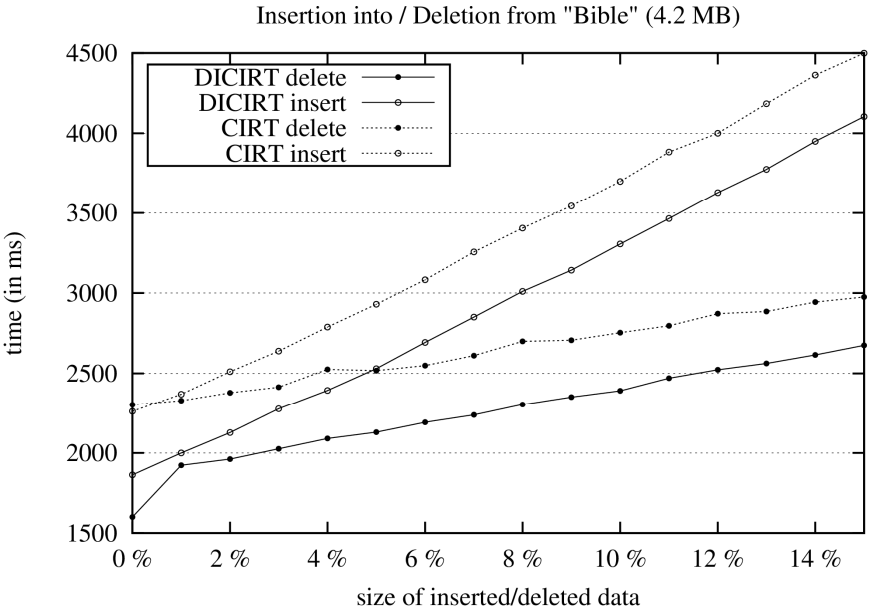


Fig. 8. Comparison of DICIRT and CIRT for deleting from and inserting into "Bible"

4 Summary and Conclusions

Direct search and modification of e.g. the n^{th} word in compressed texts without an external index and without prior decompression may be of significant advantage for applications like big data collections, column-oriented databases, main memory

databases, XML compression, and data exchange with mobile client devices. However, direct search and modification of the n^{th} word in compressed texts is not possible for common text compressors like BWT without an extra index or prior decompression. The Compressed Indexed Reversible Transformation (CIRT) [1] allows for fast search operations, but only for slow insert and delete operations directly on the transformed text, because for insert or delete operations, CIRT has to be decompressed back to IRT before the insert or delete operation and compressed again to CIRT afterwards.

We have developed DICIRT, an approach based on CIRT that besides fast search operations also supports fast insert and delete operations directly on the compressed text, i.e., no decompression to IRT or even back to the uncompressed text *S* is required. Our experimental results have shown that CIRT is comparable in compression ratio and speed to the approach of [1] and also comparable to bzip2. Insertions using DICIRT on CIRT(*S*) up to a size of 8% of the size of *S* are faster than decompressing CIRT(*S*) to *S* using bzip2, followed by compressing *S* back to CIRT(*S*) using bzip2 again. Furthermore, deletions using DICIRT on CIRT(*S*) up to a size of even 15% of the size of *S* are faster than decompressing CIRT(*S*) to *S* followed by compressing *S* to CIRT(*S*) when using bzip2 again. To summarize, we consider DICIRT as a useful technique for fast insertion and deletion operations on compressed texts, which can be used e.g. within XML databases or column-oriented big text databases.

References

1. Böttcher, S., Bültmann, A., Hartel, R.: Search and Modification in Compressed Texts. In: 2011 Data Compression Conference (DCC 2011), Snowbird, UT, USA, pp. 403–412 (2011)
2. Burrows, M., Wheeler, D.: A block sorting lossless data compression algorithm. Technical Report 124 (1994)
3. Buneman, P., Grohe, M., Koch, C.: Path Queries on Compressed XML. In: Proceedings of 29th International Conference on Very Large Data Bases, Berlin, Germany, pp. 141–152 (2003)
4. Zhang, N., Kacholia, V., Özsu, M.: A Succinct Physical Storage Scheme for Efficient Evaluation of Path Queries in XML. In: Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, Boston, MA, USA, pp. 54–65 (2004)
5. Böttcher, S., Hartel, R., Jacobs, T.: Fast multi-update operations on compressed XML data. In: Gottlob, G., Grasso, G., Olteanu, D., Schallhart, C. (eds.) BNCOD 2013. LNCS, vol. 7968, pp. 149–164. Springer, Heidelberg (2013)
6. Huffman, D.A.: A method for the construction of minimum-redundancy codes. In: Proceedings of the I.R.E., pp. 1098–1101 (1952)
7. Fraenkel, A., Klein, S.: Robust Universal Complete Codes for Transmission and Compression. Discrete Applied Mathematics 64, 31–55 (1996)
8. Golomb, S.: Run-length encodings. IEEE Transactions on Information Theory 12, 399–401 (1966)
9. Witten, I., Neal, R., Cleary, J.: Arithmetic Coding for Data Compression. Commun. ACM 30, 520–540 (1987)

10. Martin, G.N.N.: Range encoding: an algorithm for removing redundancy from a digitized message. In: Video and Data Recording Conference, Southhampton (1979)
11. Ziv, J., Lempel, A.: A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory 23, 337–343 (1977)
12. Ziv, J., Lempel, A.: Compression of Individual Sequences via Variable-Rate Coding. IEEE Transactions on Information Theory 24, 530–536 (1978)
13. Welch, T.: A Technique for High-Performance Data Compression. IEEE Computer 17, 8–19 (1984)
14. Cleary, J., Witten, I.: Data Compression Using Adaptive Coding and Partial String Matching. IEEE Transactions on Communications 32, 396–402 (1984)
15. Cormack, G., Horspool, R.: Data Compression Using Dynamic Markov Modelling. Comput. J. 30, 541–550 (1987)
16. Nevill-Manning, C., Witten, I.: Identifying Hierarchical Structure in Sequences: A Linear-Time Algorithm. J. Artif. Intell. Res. (JAIR) 7, 67–82 (1997)
17. Kreft, S., Navarro, G.: LZ77-Like Compression with Fast Random Access. In: 2010 Data Compression Conference (DCC 2010), Snowbird, UT, USA, pp. 239–248 (2010)
18. Bille, P., Landau, G., Raman, R., Sadakane, K., Satti, S., Weimann, O.: Random Access to grammar-Compressed Strings. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, pp. 373–389 (2011)
19. Chan, H.-L., Hon, W.-K., Lam, T., Sadakane, K.: Compressed indexes for dynamic text collections. ACM Transactions on Algorithms 3 (2007)
20. Sadakane, K.: Succinct data structures for flexible text retrieval systems. J. Discrete Algorithms 5, 12–22 (2007)
21. Ferragina, P., Manzini, G.: Indexing compressed text. J. ACM 52, 552–581 (2005)
22. Ferragina, P., Manzini, G.: An experimental study of an opportunistic index. In: Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, Washington, DC, USA, pp. 269–278 (2001)
23. Salson, M., Lecroq, T., Léonard, M., Mouchard, L.: A four-stage algorithm for updating a Burrows-Wheeler transform. Theor. Comput. Sci. 410, 4350–4359 (2009)
24. Léonard, M., Mouchard, L., Salson, M.: On the number of elements to reorder when updating a suffix array. J. Discrete Algorithms 11, 87–99 (2012)
25. Mäkinen, V., Navarro, G.: Succinct Suffix Arrays based on Run-Length Encoding. Nord. J. Comput. 12, 40–66 (2005)
26. Grossi, R., Gupta, A., Vitter, J.: High-order entropy-compressed text indexes. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, USA, pp. 841–850 (2003)

MXML Path-Based Storage and Ordered-Based Context Manipulation

Nikolaos Fousteris[1,⋆], Manolis Gergatsoulis[1], and Yannis Stavrakas[2]

[1] Database & Information Systems Group (DBIS),
Laboratory on Digital Libraries and Electronic Publishing,
Department of Archives and Library Science, Ionian University,
Ioannou Theotoki 72, 49100 Corfu, Greece
{nfouster,manolis}@ionio.gr
[2] Institute for the Management of Information Systems (IMIS),
R. C. Athena,
Artemidos 6 & Epidavrou, Marousi 15125, Greece
yannis@imis.athena-innovation.gr

Abstract. The problem of storing and querying XML data using relational databases has been considered a lot and many techniques have been developed. MXML is an extension of XML suitable for representing data that assume different facets, having different value or structure under different contexts, which are determined by assigning values to a number of dimensions. We currently work on the problem of converting MXML to SQL queries, in order to be executed over MXML data stored in relational databases. As part of this work, in this paper, we explore a path-based technique for storing MXML documents in relational databases, based on similar techniques previously proposed for conventional XML documents. Also, we present a new ordered-based approach on representing context in such a way so as to facilitate the formulation of context-aware queries, and we show how we can manipulate context.

1 Introduction

Multidimensional XML (MXML) [3] is an extension of XML, which allows context specifiers to qualify element and attribute values, and specify the contexts under which the document components have meaning. MXML is therefore suitable for representing data that assume different facets, having different value or structure, under different contexts.

In order to express context-aware queries over MXML data, we use the MXPath (Multidimensional XPath) language [9], which is an extension of conventional XPath, suitable for navigating in MXML documents.

The main contribution of this paper is to provide efficient tools and methods, which are necessary for the conversion of MXPath queries to SQL queries over a relational schema, that is used for storing MXML documents. For this

⋆ Supported by State Scholarships Foundation of Greece (IKY), Makri 1 and Dionisiou Areopagitou 117 42 Athens - Greece.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 203–212, 2013.

reason, we first propose a path-based technique for storing MXML documents in relational databases. According to this technique the constructed relational schema contains several tables, each of them storing a different type of nodes of the MXML-tree. Also, we use additional tables to represent context and all possible paths of a MXML-tree in a way that it can be used and manipulated by SQL queries. Furthermore, we propose techniques for context manipulation.

2 Preliminaries

In MXML [3,4], data assume different facets, having different value or structure, under different contexts according to a number of *dimensions* which may be applied to elements and attributes. The notion of "world" is fundamental in MXML. A world represents an environment under which data obtain a meaning. A *world* is determined by assigning to every dimension a single value, taken from the domain of the dimension. In MXML we use syntactic constructs called *context specifiers* that specify sets of worlds by imposing constraints on the values that dimensions can take. The elements/attributes that have different facets under different contexts are called *multidimensional elements/attributes*. Each multidimensional element/attribute contains one or more facets, called *context elements/attributes*, accompanied with the corresponding context specifier which denotes the set of worlds under which this facet is the holding facet of the element/attribute. The syntax of MXML is shown in Example 1, where a MXML document containing information about a book is presented.

Example 1. The MXML document shown below represents a book in a book store. Two dimensions are used namely `edition` whose domain is {`greek, english`}, and `customer_type` whose domain is {`student, library, teacher`}.

```
<book isbn=[edition=english]"0-13-110362-8"[/] [edition=greek]"0-13-110370-9"[/]>
  <title>The C programming language</title>
  <authors>
     <author>Brian W. Kernighan</author>
     <author>Dennis M. Ritchie</author>
  </authors>
  <@publisher>
     [edition = english] <publisher>Prentice Hall</publisher>[/]
     [edition = greek] <publisher>Klidarithmos</publisher>[/]
  </@publisher>
  <@translator>
     [edition = greek] <translator>Thomas Moraitis</translator>[/]
  </@translator>
  <@price>
     [edition=english]<price>15</price>[/]
     [edition=greek,customer_type in {student, teacher}]<price>9</price>[/]
     [edition=greek,customer_type=library]<price>12</price>[/]
  </@price>
  <@cover>
     [edition=english]<cover><material>leather</material></cover>[/]
     [edition=greek]
        <cover>
           <material>paper</material>
           <@picture>
             [customer_type=student]<picture>student.bmp</picture>[/]
```

```
            [customer_type=library]<picture>library.bmp</picture>[/]
          </@picture>
        </cover>
      [/]
  </@cover>
</book>
```

Notice that multidimensional elements are the elements whose name is preceded by the symbol `@` while the corresponding context elements have the same element name but without the symbol `@`.

3 Properties of MXML Documents

3.1 Graphical Model of MXML and Node Indexing

In this section, we present a graphical model for MXML called *MXML-tree*. The proposed model is node-based and each node is characterized by a unique "id".

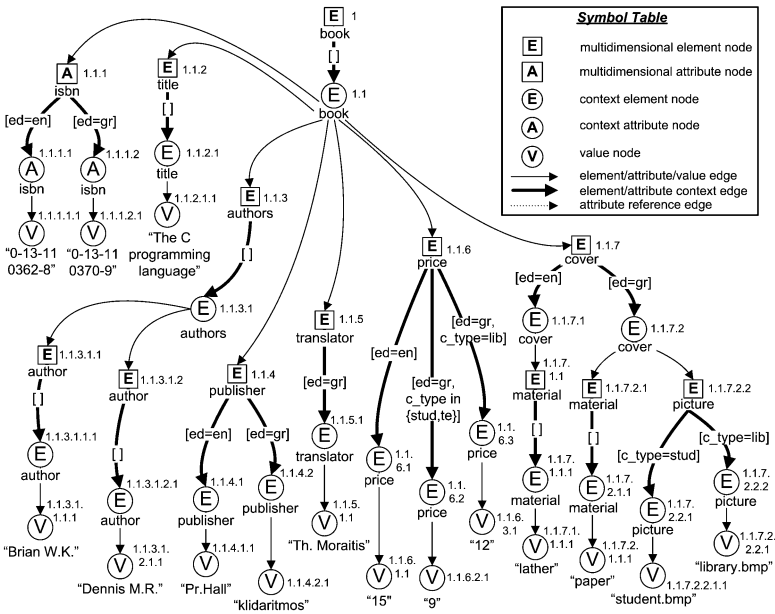


Fig. 1. Graphical representation of MXML (MXML tree)

In MXML-tree, there are the following node types: *multidimensional element nodes*, *context element nodes*, *multidimensional attribute nodes*, *context attribute nodes*, and *value nodes*. The *context element nodes*, *context attribute nodes*, and *value nodes* correspond to the element nodes, attribute nodes and value nodes in a conventional XML tree. Each multidimensional/context element

node is labelled with the corresponding element name, while each multidimensional/context attribute node is labelled with the corresponding attribute name. As in conventional XML, value nodes are leaf nodes and carry the corresponding value. The facets (context element/attribute nodes) of a multidimensional node are connected to that node by edges labelled with context specifiers denoting the conditions under which each facet holds. These edges are called *element/attribute context edges* respectively. Context elements/attributes are connected to their child elements/attribute or value nodes by edges called *element/attribute/value edges* respectively.

For indexing the nodes of a MXML tree, we use the Dewey labelling schema [8]. In general, this schema assigns to each node a dotted format identification number, according to the hierarchical position (level, sibling number) of that node in the MXML tree. So, assuming that $N_1, N_2, \ldots, N_d$ is a sequence of nodes contained in a path of the MXML tree, such that N_1 is the root node and N_{i-1} is the parent node of node N_i, we define the Dewey labelled index of node N_d denoted as the dotted format identification number $s_{N_1}.s_{N_2}.s_{N_3}\ldots s_{N_d}$, where s_{N_i} is the position number of node N_i among its siblings.

Example 2. In Fig. 1, we see the representation of the MXML document of Example 1 as a MXML-tree and the dotted format of dewey indexing. For saving space, in Fig. 1 we use obvious abbreviations for dimension names and values that appear in the MXML document.

3.2 Properties of Contexts

Context specifiers qualifying element/attribute context edges give the *explicit contexts* of the nodes to which these edges lead. The explicit context of all the other nodes of the MXML-tree is considered to be the *universal context* [ ], denoting the set of all possible worlds. The explicit context can be considered as the true context only within the boundaries of a single multidimensional element/attribute. When elements and attributes are combined to form a MXML document, the explicit context of each element/attribute does not alone determine the worlds under which that element/attribute holds, since when an element/attribute e_2 is part of another element e_1, then e_2 have substance only under the worlds that e_1 has substance. This can be conceived as if the context under which e_1 holds is inherited to e_2. The context propagated in that way is combined with (constraint by) the explicit context of a node to give the *inherited context* for that node. Formally, the inherited context $ic(q)$ of a node q is defined as $ic(q) = ic(p) \cap^c ec(q)$, where $ic(p)$ is the inherited context of its parent node p and $ec(q)$ is the explicit context of node q. $\cap^c$ is an operator called *context intersection* defined in [7] which combines two context specifiers and computes a new context specifier which represents the intersection of the worlds specified by the original context specifiers. The evaluation of the inherited context starts from the root of the MXML-tree. By definition, the inherited context of the root of the tree is the universal context.

As in conventional XML, the leaf nodes of MXML-trees must be value nodes. The *inherited context coverage* of a node further constraints its inherited context, so as to contain only the worlds under which the node has access to some value node. The inherited context coverage $icc(n)$ of a node n is defined as follows: if n is a leaf node then $icc(n) = ic(n)$; otherwise $icc(n) = icc(n_1) \cup^c icc(n_2) \cup^c ... \cup^c icc(n_k)$, where $n_1, \ldots, n_k$ are the child element nodes of n. $\cup^c$ is an operator called *context union* defined in [7] which combines two context specifiers and computes a new one which represents the union of the worlds specified by the original context specifiers.

4 Storing MXML in Relational Databases

4.1 Path-Based Relational Schema

In this section, we present a path-based technique for storing MXML documents using relational databases, which extends the technique presented in [10] for the case of XML. In our path-based relational schema, we use an Element Table, an Attribute Table and a Value Table for storing element, attribute and value nodes respectively. Each one of these tables contains the id of each node (node id) based on the Dewey-labelling schema and the id (path id) representing the path in the Path Table, which leads to that node. For the Value Table, the value of the node is additionally stored. On the other hand, the Path Table contains all possible paths of the MXML document starting from the root node, assigning to each path a unique id. As we show in the following example, the paths stored in the Path Table are path expressions with two more additional characteristics and we call them *simple path expressions*. The first one is the "$->$" notation used in MXPath [9] for indicating multidimensional nodes and the second one is the character "`#`", which is added before every "`/`" separator for helping through the MXPath to SQL query conversion. Finally, as described in sub-section 4.2, relational schema contains tables for storing explicit context or inherited context coverage in a binary-based format (world vectors).

Example 3. Fig. 2, depicts (parts of) the tables contained in the Path-based relational schema, for storing the MXML tree of Fig. 1.

Element Table	
node_id	path_id
1	1
1.1	2
1.1.2	5
....	....

Attribute Table	
node_id	path_id
1.1.1	3
1.1.1.1	4
1.1.1.2	4

Value Table		
node_id	path_id	value
1.1.1.1.1	4	0-..-8
1.1.1.2.1	4	0-..-9
....	....	....

Path Table	
path_id	path
1	#/->book
2	#/book
3	#/book#/->@isbn
4	#/book#/@isbn
5	#/book#/->title
....	....

Fig. 2. The Path-based Relational Schema

4.2 Ordered-Based Context Representation

We use an Ordered-Based technique to store the context in such a way so as to facilitate the formulation of context-aware queries, by reducing the size of tables and the number of joins. The basic idea behind this technique is that we achieve the total ordering of all possible worlds based on a) a total ordering of dimensions and b) a total ordering of the elements in the domain of each dimension. So, for k dimensions with each dimension i having m_i possible values, we may have $n = m_1 * m_2 * \ldots * m_k$ possible ordered worlds. Each of these worlds is assigned a unique integer value between 1 and n.

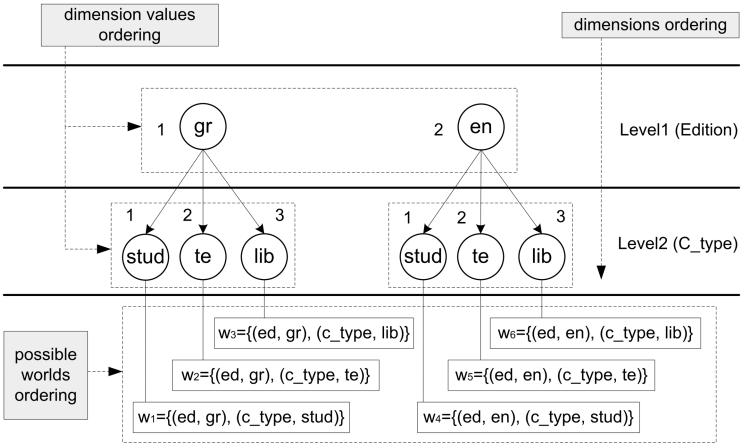


Fig. 3. Possible Worlds Ordering

Example 4. In Fig. 3, we present how we can order all possible worlds of Example 1 according to the dimensions and the dimension values. In order to show this ordering, we use a forest of trees. As we can see, each dimension of the MXML document corresponds to a level in the forest. The ordering of these levels represents the ordering of dimensions. Also, for each level we can see the ordering of all possible values of the related dimension, under each node of the previous level. Each possible world can be produced by traversing a path from a root node of the forest to a leaf node of the corresponding tree. Finally, the order of the forest's leaves represents the total ordering of all possible worlds assigning a unique integer to each world $(w_1, w_2, \ldots, w_6)$.

Assuming that all possible worlds of a MXML document are totaly ordered, we define a vector of binary digits called World Vector.

Definition 1. *Given a total ordering of worlds $W = (w_1, w_2, \ldots, w_n)$, where n is the number of possible worlds, we define as $V(c) = (a_1, a_2, \ldots, a_n)$ the World Vector of a context specifier c, where a_i with $i = 1, 2, \ldots, n$, is a one bit value*

containing 1 *if the world* w_i *is between the worlds represented by* c *or* 0 *if* w_i *is not included in the worlds represented by* c.

Fig. 4 shows how we can store dimensions' information to a relational schema. One table (Table D) is used for storing ordered dimensions and one separate table D_i with $i = 1, 2, \ldots, k$ is used for storing the ordered values $d_{i,j}$ with $j = 1, 2, \ldots, m_i$ and m_i is the number of the different values of dimension D_i.

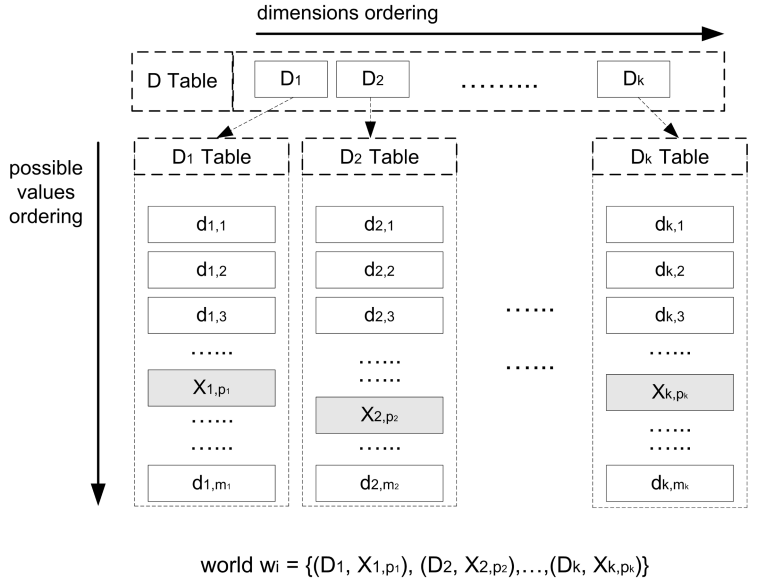


Fig. 4. Ordered-Based Representation in Relational Schema

Finding the Position of a World in a World Vector. Using the Ordered-Based Representation to represent worlds, the problem of defining the position corresponding to a specific world in a world vector arises.

As shown in Fig. 4, assuming that a context specifier contains the world w_i, and each number $p_1, p_2, \ldots, p_k$ represents the position of a value among the ordered values of each dimension $D_1, D_2, \ldots, D_k$ respectively, we can find the bit-position i corresponding to this world in the world vector of the context specifier, using the following formula:

$$i = p_k + \sum_{j=2}^{k}[(p_{j-1} - 1) * (\prod_{w=j}^{k} m_w)]$$

Example 5. Fig. 5, depicts (parts of) the Explicit Context Table, and the Inherited Coverage Table obtained by encoding the context information appearing in the MXML-tree of Fig. 1. Also, we can see the contents of the tables D, D_1 and

D Table	
dimension_id	dimention_name
1	edition
2	customer_type

D_1 Table	
value_id	value
1	greek
2	english

D_2 Table	
value_id	value
1	student
2	teacher
3	library

Inherited Coverage Table	
node_id	world_vector
1.1	111111
1.1.1	111111
1.1.1.1	000111
1.1.1.1.1	000111
....	....

Explicit Context Table	
node_id	world_vector
1.1	111111
1.1.1	111111
1.1.1.1	000111
....	....
1.1.6.3	001000
....	....

Fig. 5. Context Tables

D_2 containing the ordering information for all possible worlds. For example, the explicit context of the node with `node_id=1.1.1.1` includes the worlds:

$w'_1 = \{$`(edition, english)`, `(customer_type, student)`$\}$,
$w'_2 = \{$`(edition, english)`, `(customer_type, teacher)`$\}$ and
$w'_3 = \{$`(edition, english)`, `(customer_type, library)`$\}$

According to the ordering of Fig. 3, the bit-positions of these worlds in the world vector are 4, 5 and 6 respectively. As a result, the explicit context specifier of the node is encoded in the Explicit Context Table as one row with `node_id=1.1.1.1` and the world vector 000111.

Finding the World Corresponding to a Bit in a World Vector. The opposite problem of finding the position of a world in a world vector is the problem of finding which world corresponds to a bit-position i of a world vector. In order to achieve this, we can use the following algorithm expressed as a function, using the notation of Fig. 4.

Function Convert_bit-position_i_of_world_vector_to_world(i)
Input: The position i of a bit in a world vector, **Output:** A world $(p_1, p_2, ..., p_k)$
begin
If (i>1) **then**
k'=1; i'=i-1;
While (k'<k) **do**
$a_{k'}$=i' DIV ($m_{k'+1}$*$m_{k'+2}$*...*m_k); $u_{k'}$=i' MOD ($m_{k'+1}$*$m_{k'+2}$*...*m_k);
If ($a_{k'}$=0) **then** $p_{k'}$=1; **else** $p_{k'}$=$a_{k'}$+1; **end if**
k'=k'+1; i'=$u_{k'}$;
end while
If (i'=0) **then** p_k=1; **else** p_k=i_k+1; **end if**
else
p_1=p_2=...=p_k=1;
end if
end

The above algorithm, takes as input the i position of a world in a world vector. The output of the algorithm is a sequence of numbers $(p_1, p_2, \ldots, p_k)$. Each number p_i represents the position of a value among the ordered values of dimension D_i. Using this position, we get the value X_{i,p_i} of the dimension D_i from the appropriate table D_i of Fig. 4. The set of pairs (D_i, X_{i,p_i}) represents the resulting world.

5 Ordered-Based Context Operations and Comparison

During MXPath to SQL query transformation, it is necessary to transfer constrains posed by context specifiers in MXPath queries to SQL conditions. To achieve that, we need a way to compare set of worlds expressed by context specifiers. In this section we show how we can apply set operations and comparison among context specifiers when they are represented in Ordered-Based Context Representation.

We first demonstrate how the intersection and union of context specifiers is performed at the level of World Vectors.

Lemma 1. *Let c_1, c_2 be two context specifiers and b_1, b_2 the world vectors of c_1, c_2 respectively. Then the world vector b_3 of the context intersection $c_1 \cap^c c_2$ is obtained by applying the AND operation[1] to the corresponding bits of b_1 and b_2. Respectively, the world vector b_4 of the context union $c_1 \cup^c c_2$ is obtained by applying the OR operation[2] to the corresponding bits of b_1 and b_2.*

It is also possible to compare two context specifiers using their world vectors. This is very useful when we are trying to transform MXML queries containing relevant conditions to SQL queries over a Relational Database. These conditions imply comparisons between the context specifiers which are stored with the MXML document in the relational schema, and the context specifiers which are used in the MXML queries. Similarly to AND_b and OR_b, in Lemma 2 we use the abbreviation XOR_b for the bit-wise XOR operation.

Lemma 2. *Let c_1, c_2 be two context specifiers and b_1, b_2 the world vectors of c_1, c_2 respectively. Then*

1. *$c_1 = c_2$ iff $b_1 = b_2$, alternatively $c_1 = c_2$ iff $(b_1\ XOR_b\ b_2) = 0$*
2. *$c_1 \neq c_2$ iff $NOT(b_1 = b_2)$*
3. *$c_1 \geq c_2$ iff $(b_1\ AND_b\ b_2) = b_2$*
4. *$c_1 > c_2$ iff $((b_1\ AND_b\ b_2) = b_2)$ and $(b_1 \neq b_2)$.*

Example 6. Consider the context specifiers:
$c_1 = [edition = english]$ and
$c_2 = [edition = english, customer\_type = student]$.
Calculating the world vectors of those two context specifiers we have $V(c_1)$= 000111=b_1 and $V(c_2)$=000100=b_2. Then the expression $c_1 \geq c_2$ is *true*, as $(b_1\ AND_b\ b_2) = (000111\ AND_b\ 000100) = 000100 = b_2$ (see Case 3 of Lemma 2).

[1] For this bit-wise AND operation we will use the abbreviation AND_b.
[2] For the bit-wise OR operation we will use the abbreviation OR_b.

6 Discussion

The problem of storing and querying XML data in relational databases has been intensively investigated [1,10,6,2,8,5] during the last years. Extending this problem for MXML, this paper presented a path-based technique for storing MXML documents in relational schemas. According to this technique, path information of the MXML document is included in the relational schema for better querying performance, because of reduced sql joins. Furthermore, we presented a context representation technique for storing context in a relational database. Based on this schema, we explained how we can manipulate context and how we can perform operations and comparisons between context specifiers in order to transfer context constrains from MXML queries expressed in MXPath to SQL conditions. Using all the above features, we currently work on a system, which using the appropriate algorithms, will be able to store MXML data in relational databases, transform MXPath to SQL queries and derive useful experimental results concerning querying performance and efficiency.

References

1. Deutsch, A., Fernandez, M.F., Suciu, D.: Storing Semistructured Data with STORED. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 431–442. ACM Press (1999)
2. Du, F., Amer-Yahia, S., Freire, J.: ShreX: Managing XML Documents in Relational Databases. In: Proc. of VLDB 2004, pp. 1297–1300. Morgan Kaufmann (2004)
3. Gergatsoulis, M., Stavrakas, Y., Karteris, D.: Incorporating Dimensions in XML and DTD. In: Mayr, H.C., Lazanský, J., Quirchmayr, G., Vogel, P. (eds.) DEXA 2001. LNCS, vol. 2113, pp. 646–656. Springer, Heidelberg (2001)
4. Gergatsoulis, M., Stavrakas, Y., Karteris, D., Mouzaki, A., Sterpis, D.: A Web-Based System for Handling Multidimensional Information through MXML. In: Caplinskas, A., Eder, J. (eds.) ADBIS 2001. LNCS, vol. 2151, pp. 352–365. Springer, Heidelberg (2001)
5. Ramanath, M., Freire, J.-L., Haritsa, J.R., Roy, P.: Searching for efficient XML-to-relational mappings. In: Bellahsène, Z., Chaudhri, A.B., Rahm, E., Rys, M., Unland, R. (eds.) XSym 2003. LNCS, vol. 2824, pp. 19–36. Springer, Heidelberg (2003)
6. Shanmugasundaram, J., Shekita, E.J., Kiernan, J., Krishnamurthy, R., Viglas, S., Naughton, J.F., Tatarinov, I.: A General Technique for Querying XML Documents using a Relational Database System. SIGMOD Record 30(3), 20–26 (2001)
7. Stavrakas, Y., Gergatsoulis, M.: Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 183–199. Springer, Heidelberg (2002)
8. Tatarinov, I., Viglas, S., Beyer, K.S., Shanmugasundaram, J., Shekita, E.J., Zhang, C.: Storing and querying ordered XML using a relational database system. In: Proc. of the 2002 ACM SIGMOD Int. Conf. on Management of Data, pp. 204–215. ACM (2002)
9. Fousteris, N., Stavrakas, Y., Gergatsoulis, M.: Multidimensional XPath. In: Proc. of iiWAS 2008, pp. 162–169 (2008)
10. Yoshikawa, M., Amagasa, T., Shimura, T., Uemura, S.: XRel: a path-based approach to storage and retrieval of XML documents using relational databases. ACM Transactions on Internet Technology 1(1), 110–141 (2001)

Processing k Nearest Neighbor Queries for Location-Dependent Data in MANETs

Yuka Komai, Yuya Sasaki, Takahiro Hara, and Shojiro Nishio

Department of Multimedia Engineering Graduate School of Information Science and Technology, Osaka University, Yamadaoka 1-5, Suita-shi, Osaka, Japan

Abstract. KNN queries, which retrieve the k nearest data items associated with a location (location-dependent data) from the location of the query issuer, are available for location-based services (LBSs) in mobile environments. Key challenges in designing system protocols for MANETs include low-overhead adaptability to network topology changes due to node mobility, and query processing that achieves high accuracy of the query result without a centralized server. In this paper, we propose the Filling Area (FA) method to process a kNN query in MANETs. To achieve a small search area, data items remain at nodes near the locations with which the items are associated, and nodes cache data items whose locations are near their own. When a node issues a query, neighboring nodes send back their copies, which will likely include the query result.

Keywords: MANET, kNN query, location-dependent data, LBS.

1 Introduction

Recently, there has been an increasing interest in *mobile ad hoc networks* (*MANETs*), which are composed of only mobile nodes. In MANETs, all mobile nodes play the role of a router. Even if the source and the destination mobile nodes are not within communication range of each other, data packets are forwarded to the destination mobile node by relaying the transmission through intermediary mobile nodes. Since no special infrastructure is required, many MANET-based applications are expected to be developed in various fields such as military affairs and rescue operations.

A location-based service (LBS) is a typical application for MANETs. In an LBS, it is common for a node to issue queries in search of information on a specific location, which is retained by mobile nodes. In such a case, it is effective to process the queries as k nearest neighbor (kNN) queries, which search the kNNs, i.e., k nearest data items associated with a location (location-dependent data), from the specified location (*query point*).

MANETs possess notably different characteristics from wired networks and wireless sensor networks, such as limitations on network bandwidth, and dynamic topology change due to the movement of mobile nodes. In addition, as there is

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 213–227, 2013.

no centralized server in MANETs, existing kNN processing methods based on a centralized server are not applicable to MANETs. Therefore, key challenges in designing system protocols for MANETs include low-overhead adaptability to network topology changes due to node mobility, and query processing that achieves high accuracy of the query result without a centralized server.

In this paper, we propose the Filling Area (FA) method, to process a kNN query which searches for the k nearest location-dependent data items, in order to achieve low traffic and high accuracy of the query result in MANETs. In the FA method, when a node issues a kNN query, it acquires data items retained by nodes within a specific region (*search area*), which is a circular area whose center point is its own location [1]. Nodes reply with data items (original and cached) which will likely include the query result, avoiding replies of the same data item by overhearing messages. To achieve a small search area, data items remain at nodes near the locations with which the items are associated, and nodes cache data items (as copies) whose locations are near their own. In particular, when a node which retains data items moves beyond a given *data boundary*, away from the location with which they are associated, it forwards the items to the nodes nearest to the location with which the items are associated. We also show some experimental results to verify that our method can reduce traffic compared with the existing method and also achieve high accuracy of the query result.

The remainder of this paper is organized as follows. In section 2, we introduce related works. In section 3, we present our proposed kNN query processing method. In section 4, we show the results of the simulation experiments. Finally, we summarize this paper in section 5.

2 Related Work

2.1 Location-Dependent Data Management in LBS

In [3], the authors proposed a line-based data dissemination protocol for sensor networks. Sinks disseminate data items to nodes within a vertical virtual line which divides the field into two parts, and nodes search for data items there. This method does not assume kNN query processing and replication of location-dependent data. In [11], the authors proposed the Skip-Copy method, specifically aimed at managing location-dependent data items in MANETs. This method sparsely distributes copies of location-dependent data items, and nodes access a single location-dependent data item, in contrast to our method's aim of acquiring multiple location-dependent data items near a specific point. In [13], the authors proposed a system which provides an LBS which does not depend on pre-established infrastructure. The system uses a mobile agent that remains within a certain geographical area, moving among mobile nodes in the area. Our method is inspired from this approach in localizing data items.

[1] In this paper, the query point is the location of the query-issuing node. We assume that the query-issuing node retrieves nearby information around itself.

2.2 kNN Query Processing

In [10], the authors proposed a method for efficiently acquiring kNNs from mobile query points. This method can reduce disk access search costs for databases, when the query point is moving and kNN results change. It assumes that the information of all static objects (such as hospitals or schools) has been previously obtained (i.e., a centralized method). In [2], the authors defined Continuous All k-Nearest Neighbor (CAkNN) queries which continuously identify all nodes' closest neighboring nodes, and proposed a 'Proximity' algorithm for efficiently processing such queries in smartphone networks. This algorithm only works well in areas covered by a set of network connectivity points (e.g., cellular towers for cellular networks). Moreover, each node must regularly report its positional information to the query processor. In [12], the authors proposed a method to continuously monitor kNNs in a wireless sensor network. Here, sensors detect objects moving around the target region, and these sensors collaborate to continuously monitor the kNNs (k nearest objects) from the query point. However, since this method assumes that the sensors are statically deployed, it is not applicable to MANETs. In [5], we proposed a kNN query processing method for MANET. In this method, a node floods a query to nodes within a specific circular region centered on the query point, and each node receiving the query replies with information on itself. Thus, this method can avoid flooding in the entire network. However, the search object in this method is the k nearest nodes (not location-dependent data items). In [7], the authors proposed a kNN query processing method for a pure mobile P2P environment (i.e., MANET). Here, nodes collaborate in answering the query, and acquire kNNs with less communication cost than that of centralized methods. This work addresses a similar problem to that of the present study, but in the former, k is less, and in the latter more, than the storage capacity of the nodes.

3 KNN Query Processing Method

In this section, first we describe the design policy of our proposed method and assumptions. Then, we describe how data items are managed in the FA method. Finally, we describe in detail how kNN queries are processed in the FA method.

3.1 Design Policy

In MANETs, it is highly important to minimize traffic, due to limitations on network bandwidth and the batteries of the mobile nodes. The periodic broadcast of beacon messages by nodes, even when no node is searching kNNs in the network, causes unnecessary traffic; and thus a beacon-less method is more suitable for MANETs. However, without exchanging beacon messages, nodes cannot know neighboring nodes' information beforehand. Therefore, we have designed a beacon-less method involving on-demand searches.

Since there is no centralized server in a MANET, a centralized approach is inappropriate. For example, if data items were transmitted to a specific node

as soon as they were generated, the query-issuing node could acquire kNNs by transmitting a query to the specific node because the specific node would know the kNNs for all the data items. However, this is not realistic owing to the nodes' limited storage capacity. Moreover, the transmission of numerous queries to the specific node would cause a temporary spike in traffic volume in the vicinity of the node. Therefore, we have designed a distributed kNN processing method.

In an LBS, users often require location-dependent data items from near their own location; and when using a kNN query, they repeatedly require the kNNs from their own location. Therefore, a method is needed in which nodes retain data items whose location is close to their own, keeping in mind that nodes' locations will change as they move. Note that kNNs from the own location change along node mobility.

Since mobile nodes make great demands on the limited communication bandwidth in data transmission, packet losses and packet retransmissions may occur when the network is congested, resulting in information failing to be transmitted. Thus, the amount of information transmitted by mobile nodes should be minimized. It is particularly wasteful to flood a message over the entire network. Thus, in our methods, the query-issuing first determines the search area as small as possible maintaining the accuracy of the query result, and searches within it.

3.2 Assumptions

The system environment is assumed to be a MANET in which all mobile nodes have the same radio communication facility, whose communication range is a circle of a fixed size. We assume that the MANET is sufficiently dense that network partitioning does not occur. In the MANET, mobile nodes retrieve location-dependent data items on mobile nodes by using kNN queries.

We assign a unique *node identifier* to each mobile node in the system. The set of all mobile nodes in the system is denoted by $M = \{ M_1, M_2, \cdots, M_n \}$, where n is the total number of mobile nodes and M_i $(1 \leq i \leq n)$ is a node identifier. Each mobile node moves freely and knows its current location (using a positioning system such as GPS). We assign a unique *data identifier* to each data item in the system. The set of all data items in the system is denoted by $D = \{ D_1, D_2, \cdots, D_m \}$, where m is the total number of data items and D_j $(1 \leq j \leq m)$ is a data identifier. Each data item includes the location information associated with the data-generating location (not skew generating pattern). For simplicity, all data items are assumed to be the same size and not updated. Nodes have a cache storage capacity of C data items.

The query-issuing node transmits a query message associated with its own location as the query point, and acquires the k nearest location-dependent data items from the query point, among all data items in the entire network.

3.3 Management of Data Items

As described above, since users repeatedly require the kNNs from their own location using kNN queries, it is useful for nodes to retain the CNNs from their

locations. In such a situation, if k is less than C, a node can know the kNNs among its cached data items without query processing. Even if k is more than C, a node often can acquire kNNs by seeking data items from only nearby nodes, because these nodes hold nearby data items.

In MANETs, it is a considerable challenge to maintain current CNNs, because the location of a given node (in this case the query point) changes due to node mobility, resulting in changes in the CNNs. If nodes frequently exchange messages including data items, they can update the CNNs from their new location, but this causes frequent packet losses and high energy consumption. On the other hand, when nodes do not frequently exchange messages, their CNN knowledge is greatly reduced and inquiries may not work well.

In the FA method, when a node generates a data item, it signals the occurrence of the item by flooding a message, including the item, over a specific area, which should be appropriately determined such that the data item is surely sent to nodes which retain it as a CNN, but not sent to an unnecessary wide area. After this, the nodes retain and cache the original data item.

Original Data Item: When a node retaining an original data item moves beyond a given *data border*, d, away from the *data point* with which the item is associated, the original item is transmitted to the node nearest that data point, using our geo-routing method (an extension of the protocol proposed in [4]), which adopts a three-way handshake protocol to send a data item to the neighboring node nearest the data point. By repeating this procedure, the data is forwarded to the node nearest its dependent location.

More specifically, in our geo-routing method, the node which retains a data item first broadcasts a *neighbor search message* when it moves beyond d. Then, when a node receives the neighbor search message, if it is closer to the data point than the source node, it sets a waiting time for sending a reply. Because nodes closer to the data point transmit a reply message after a shorter waiting time, the nearest neighboring node from the data point is the first to transmit a reply message to the source node. The source node that has received reply messages from its neighbors sends a (forwards the) *data forwarding message*, including the data item, only to the node that sent the first reply. The node that receives the data forwarding message broadcasts a neighbor search message using the same procedure. Finally, if a node that has sent a neighbor search message receives no reply messages when the data point is within its communication range, it recognizes itself as the nearest node from the data point (Fig. 1).

Therefore, a node which retains a data item can forward the item to the nearest node from the data point with low traffic because this geo-routing method requires neither the use of beacon messages nor the construction of multiple paths. By doing so, a node can acquire a data item by searching nodes from which the distance to the data point of the item is less than d.

Cached Data Item: To maintain current CNNs, a node updates its retained CNNs by exchanging messages with its neighboring nodes. Here, we define the circular area in which there are CNNs, as the node's *Assignment*, whose center

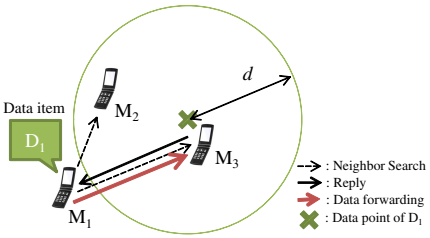


Fig. 1. Example of Forwarding an Original Data Item

point is the location at which the node updates its CNNs, and whose radius is the distance from this *update point* to the data point of the C-th nearest data item from the update point.

When a node moves farther than d from its update point, it broadcasts a request message to its neighboring nodes, which includes the IDs of data items cached by the node, and the current location of the node. Each node receiving the message sets a *waiting time* for a response based on the distance, *dist*, from its update point to the issuer of the message, using the following equation: $waiting\ time = \alpha \cdot dist$, where α is a parameter for waiting responses. The waiting time decreases, as the update point of a node nears the location of the message issuer, because a nearer node will likely retain many cached data items which the issuer will cache as CNNs. The node which has set the least waiting time selects the candidates, which are data items closer than the C-th nearest data items from the location of the message issuer, and at the expiration of its waiting time the node broadcasts a response message including the candidates. Nodes overhear response messages from each other, and if the node has no candidates which are not sent by other nodes, it does nothing, resulting in a reduction of unnecessary traffic.

3.4 Forwarding kNN Query and Replying with Result

In the FA method, when a node issues a kNN query, it floods the query to nodes within a specific region (*search area*), which is a circular area whose center point is the query point. Nodes reply with data items which will be likely included in the kNNs, avoiding replies of the same data item by overhearing messages from other nodes. Algorithm 1 shows the procedures involved in query processing. In this pseudo code, $Assignment(node)$ denotes the circular area centered on a node's update point, whose radius is the distance from that point to the furthest point among the data points of data items cached by that node; that is, in the area containing C data items with the nearest data points from the node's update point.

First, the query-issuing node specifies the requested number of kNNs, k, and determines whether its cached data items satisfy the kNNs from the query point (Line 2 in Algorithm 1). More specifically, if $k \leq C$, and the data points of the k nearest data items among data items cached by the query-issuing node are

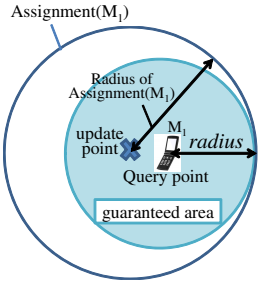


Fig. 2. Guaranteed Area

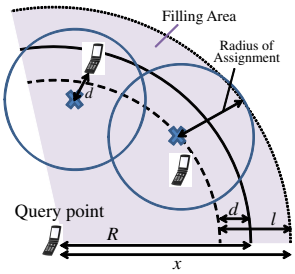


Fig. 3. Definition of R

present within the *guaranteed area* which is a circular area centered on the query point, whose radius is '*radius of Assignment(query-issuing node)—distance between the query point and its update point*', the query-issuing node can complete the search because the kNNs are guaranteed; that is, the data point of any data item outside this circular area must be farther from the query point than the data points of data items within this area (see Fig. 2). Otherwise, the query-issuing node performs query processing for acquiring kNNs in cooperation with nearby nodes.

The query-issuing node determines the radius of the search area, R, based on the information on data items cached by the query-issuing node, and k, using the following equation:

$$R = \beta \cdot l \cdot \sqrt{\frac{k}{C}} - \gamma \cdot l + d, \tag{1}$$

where l is the radius of Assignment(query-issuing node), β is a parameter to avoid misestimation of the density of data items, and γ is a parameter for reduction of the search area. According to Equation (1), R is principally determined based on the density of data items cached by the query-issuing node (i.e., $\pi l^2{:}C{=}\pi x^2{:}k$, where x is the radius of the "Filling Area", which is the circular area in which data points of the kNNs are present with high probability). Because each node caches data items, it can reply with the data items whose data points are roughly l away from its location, and thus nodes within the circular area of the inner dotted line in Fig. 3 will have data items within the Filling Area. Therefore, R can be reduced by l from x. Here, γ is basically set at 1, but since the radius of Assignment(node) is different for every node, γ may be set at less than 1 for safety. Moreover, since each node locates (at most) d away from its update point, R is enlarged by d.

Then, the query-issuing node floods the search area with a kNN query message which includes the query-issuing node's ID; the remaining requested number of kNNs, k' (i.e., k—the number of data items within the guaranteed circular area); R; the query-issuing node's assignment (i.e., its update point and l); and the query point (i.e., its own location). At the same time, the query-issuing node

Algorithm 1 Query processing

```
1:  // query-issuing node begins to search
2:  if Assignment(query-issuing node) guarantees kNNs then
3:      complete search
4:  else
5:      calculate search area
6:      send Query
7:      set WT
8:  end if

9:  // node receives Query for the first time
10: if within search area then
11:     store ID of parent
12:     store Assignment(query-issuing node)
13:     send Query
14:     set RD
15: end if

16: // node receives Reply or EmptyReply
17: store Assignment(replyNode)
18: update RD
19: if parent of source node then
20:     if query-issuing node then
21:         store data items
22:     else
23:         send Reply to parent
24:     end if
25: end if

26: // RD has passed at Node
27: if Assignment(Node) is not covered then
28:     if no data within Assignment(Node) then
29:         send EmptyReply to parent
30:     else
31:         send Reply to parent
32:     end if
33: end if

34: // WT has passed at query-issuing node
35: if k data items are acquired and area of kNNs is covered then
36:     complete search
37: else
38:     calculate new search area
39:     send Query refine
40:     set WT
41: end if
```

sets the waiting time, WT, for awaiting replies, calculated by '$\delta \cdot \lceil R/l \rceil$', where δ is a positive constant (Line 7 in Algorithm 1).

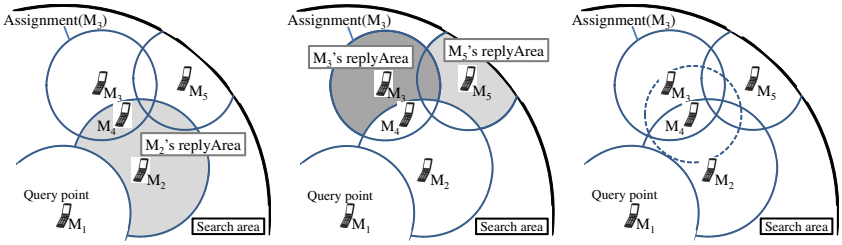
If a node receiving the query message (the receiver) is within the search area, it sets the reply delay, RD (the time from receiving a query for the first time to sending a reply), using the following equation:

$$RD = \delta \cdot lap \cdot \frac{maxArea - replyArea}{maxArea}. \tag{2}$$

$$lap = \begin{cases} 1 & (distance < l). \\ \lfloor distance/l \rfloor & (\text{otherwise}). \end{cases} \tag{3}$$

Here, $replyArea$ is an area described by 'Assignment(receiver) $\cap$ $\overline{\text{Assignment(query-issuing node)}} \cap \overline{\text{Assignment}(\forall replyNode)}$', where $replyNode$ is a node which has already issued a reply received by the receiver (e.g., in the gray-colored area in Fig. 4), $maxArea$ is a positive constant, and $distance$ is the distance from the query point to the receiver's update point. In Equation (2), RD decreases, as $distance$ decreases and $replyArea$ increases. As $replyArea$ increases, the priority of replying must be higher because more data items are present in the area with high probability, and thus the node can reply with them. Nodes within a smaller lap reply earlier every additional l, because nodes with update points closer to the query point can reply with a smaller hopcount to the query-issuing node.

A node within the search area initiates a reply message (after waiting RD), including candidate data items and its assignment, to the query-issuing node if $replyArea$ is not zero; that is, if Assignment(this node) is not fully covered by replies from other nodes (Line 27 in Algorithm 1). The candidate data items are



(a) M_2's turn (b) M_3's and M_5's turns (c) M_4's turn

Fig. 4. Example of FA Method

the (at most) k' data items which have not yet been sent in reply by other nodes, among the data items whose data points lie within the Filling Area. If the node has no candidates, it transmits an *empty reply message* to the query-issuing node, instead of a reply message. The empty reply message includes only its assignment (not data items). With the reception of the empty reply message, the query-issuing node can recognize that there are no data items in the assignment attached to a message, and thus, the method guarantees that the kNN query exhaustively searches the search area. When a node receives (or overhears) reply or empty reply messages, it recognizes the area covered by the replying node, avoiding multiple replies of the same data item; and if the overhearing and replying nodes' assignments overlap, *replyArea* is updated, resulting in an updated RD. The query-issuing node receiving a reply message updates the tentative kNN result. When WT has passed, if the number of acquired data items exceeds k and the circular area centered on the query point, whose radius is the k-nearest data point, is fully covered with assignments of all replying nodes, the search is completed and the query-issuing node updates its cached data items with the query result. If not, the query-issuing node re-estimates R (e.g., β is decreased) and repeats the same process, to avoid decreasing the accuracy of the query result; note further that there is an upper limit on the number of such repetitions (not endless).

Fig. 4 shows an example of executing query processing in which M_1 issues a kNN query, and nodes' update points correspond to their respective locations for simplicity. All nodes receiving the kNN query (M_2-M_5) set respective RDs because they are within the search area. First, M_2 initiates a reply to M_1 because its *assignmentArea* is larger and the distance between the query point and itself is less than that of the other nodes (Fig. 4(a)). Nodes overhearing M_2's reply message (M_3-M_5) store M_2's assignment, and update their respective RDs in light of this. After M_3, and then M_5, transmits a reply (empty reply) message (Fig. 4(b)), M_4 does nothing (when its own RD has passed) because its assignment is fully covered by assignments of other nodes (Fig. 4(c)).

By using this method, unnecessary transmissions of queries and replies can be suppressed, as nodes search within a limited search area and overhear the replies of other nodes.

4 Simulation Experiments

In this section, we show the results of simulation experiments evaluating the performance of our proposed method. For the simulation experiments, we used the network simulator, QualNet5.2 [9].

4.1 Simulation Model

The number of mobile nodes in the entire system is 500. These mobile nodes are present in an area of 1,000[m] × 1,000[m], and move according to the random waypoint model [1], with a movement speed and pause time of from 0.1 to v[m/sec] and 200[sec], respectively. Mobile nodes transmit messages using an IEEE 802.11b device with a data transmission rate of 11[Mbps]. The transmission power of the mobile nodes is determined such that the radio communication range is about 100[m]. Packet losses and delays occur due to radio interference. We assume that each node knows its current location.

The number of data items in the entire system is 500, and the size of data items is 128[B]. Nodes have a cache storage capacity of C data items. The query point specified by a kNN query is fixed as the location of the query-issuing node. d, α, β and γ in Equation (1), and δ in Equation (2) are respectively set to 50, 1, 1.1, 0.9, and 1 based on our preliminary experiments.

We compare the performance of the FA method with that of two alternate methods in which only original data items (not copies) remain at nodes near their respective data points in the same manner as in the FA method. The first method is the EXP method for searching location-dependent data items (extending the EXP method in [5]). We assume that each node knows the total number of data items, and the entire area size. Thus, in the EXP method, the radius of the search area, R, is determined by the following equation:

$$R = \sqrt{\frac{k \cdot area}{\pi \cdot m}} + d, \tag{4}$$

where $area$ is the area size ($area$ = 1,000×1,000), and m is the total number of data items (m = 500). The query-issuing node floods a query to nodes within the search area, and nodes receiving the query reply with data items whose data points are within the circular area centered on the query point, whose radius is 'R-d'. "EXP" in the graphs denotes this method. The other method for comparison is the FA method without maintenance of cached data items (denoted by "FAwo" in the graphs). In this method, query processing is performed in the same way as in the fuller FA method, with the exception of the cached data item maintenance described in the 'Cached Data Item' subsection of Section 3.3 above. This latter comparison demonstrates the effect of the maintenance of cached data items in the fuller FA method. In all methods, the query processing procedures are performed only once, i.e., no repetitions of a search, to evaluate the performance of methods at one time. Table 1 shows the parameters (and

their values) used in the simulation experiments. The parameters are set by default at the constant values to the left of their respective parenthetical range, and varied over this range in the simulations.

In the FA and FAwo methods, first, nodes flood data items so that others can retain them in the cache storage. After thousand seconds have passed since the simulation started, the query-issuing node, randomly chosen among all nodes, issues a kNN query. We repeat this process 1,000 times (i.e., 1,000 queries) for every 0.5 seconds, and evaluate the following two criteria.

Table 1. Parameter configuration

Symbol	Meaning	Values (Range)
C	Cache storage capacity	20 (10‘50)
k	Requested number of data items	30 (1‘60)
v	Max speed	0.5 [m/s] (0.1‘2)

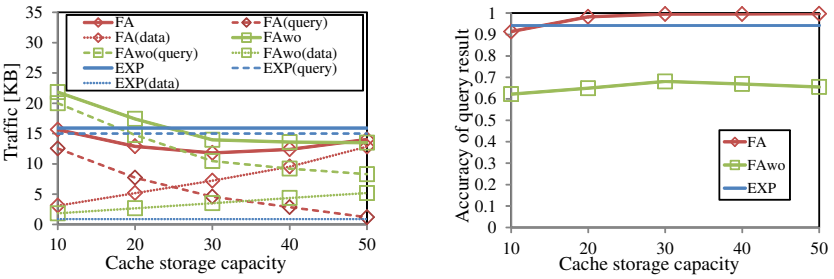
Table 2. Message types and sizes

Type	Size [B]
Neighbor search (data)	48
Reply (data)	16
Data forwarding (data)	40+128
Notification (data)	32+128
Request (data)	32
Response (data)	$24+128 \cdot t$
Query (FA and FAwo)	104
Query (EXP)	64
Reply (FA and FAwo)	$64+128 \cdot q$
Reply (EXP)	$32+128 \cdot q$
EmptyReply (FA and FAwo)	56
Ack for a reply (all)	16

– *Traffic*
We examine the total volume of query messages and replies exchanged in processing a query. Table 2 shows the size of messages in the FA method and the comparative methods, where q and t respectively denote the number of data items included in the reply, and that included in the response for the request. Here, ‘Neighbor search (data)’ denotes a neighbor search message in the maintenance process of data items described in section 3.3, and so on. We define “traffic” as the average of the total volume of messages for all queries (i.e., total volume of messages divided by 1,000).
– *Accuracy of query result*
We examine the ratio of the number of kNNs whose information is included in the kNN result acquired by the query-issuing node, to the requested number of kNNs, k. We define “accuracy of query result” as the MAP (Mean Average Precision) value, which measures the performance of the result with a ranking [6]. MAP is an average of the AP (Average Precision) of the respective queries, and AP is determined by the following equation:

$$AP_i = \frac{1}{k} \sum_{j=1}^{k} \left(\frac{g}{j} \cdot e \right), \tag{5}$$

where AP_i is AP on the i-th issued query, g is the number of data items which are included in the query result among the top-j nearest data items, and e is set as 1 if j-th nearest data item is included in the kNN result (if not, e is set as 0). MAP is the average of AP_i ($i = 1, \cdots, 1{,}000$) for 1,000 queries. Thus, MAP increases as the query-issuing node acquires the information on data items nearer to the query point.



(a) Traffic (b) Accuracy of query result

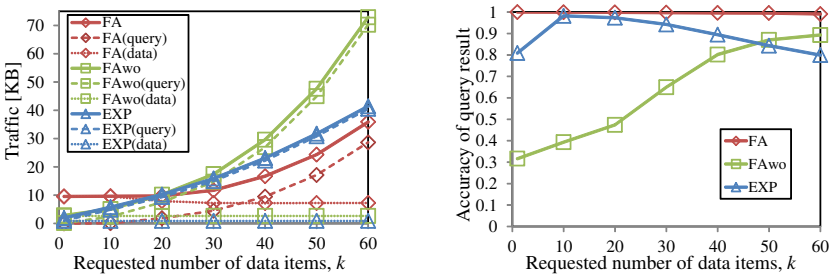
Fig. 5. Impact of Cache Storage Capacity

4.2 Impact of Cache Storage Capacity

In the FA method, the cache storage capacity affects the performance. Therefore, we first show the results of simulations when varying the capacity of cache storage, C. Fig. 5 shows the simulation results. In the graphs, the horizontal axis indicates the cache storage capacity, and the vertical axes respectively indicate the traffic in Fig. 5(a) and the accuracy of query result in Fig. 5(b). In the graphs for the traffic, '*method*(query)' and '*method*(data)' respectively denote the traffic for query processing and the traffic for data maintenance (described in section 3.3) in the *method*. Specifically, '*method*(data)' includes the traffic for maintenance of original data. 'FAwo(data)' also includes the traffic of flooding for data generation, and 'FA(data)' includes the traffic of flooding for data generation and that for cached data maintenance.

From Fig. 5(a), in the FA method, as C increases, the traffic for data maintenance increases, but the traffic for query processing decreases. This is because the number of data items exchanged for data maintenance increases, but the number of data items replied in query processing decreases (i.e., the query-issuing node has more kNNs in its cache). In the FA method without maintenance, the traffic for query processing is larger than that of the FA method because it often happens that same data items are replied from multiple nodes since the mechanism for avoiding multiple replies of same items in the FA method (i.e., overhearing and canceling replies) does not work well. This is because in the FA method without maintenance, cached data items are less relevant to their associated locations due to nodes mobility, and thus, nearby nodes cache less similar data items, but remote nodes often cache same data items, which causes multiple replies for the same items.

From Fig. 5(b), the accuracy of query result in the FA method is very high. This is because the data maintenance works well and only small number of nodes send back replies to the query-issuing node. Only when C is 10, the accuracy of query result slightly decreases, because l in Equation (1) becomes very small, and thus the query-issuing node sometimes cannot acquire k data items. In the FA method without maintenance, the accuracy of query result is low because data items cached by each node are no more the current C nearest data items



(a) Traffic (b) Accuracy of query result

Fig. 6. Impact of Requested Number of Data Items, k

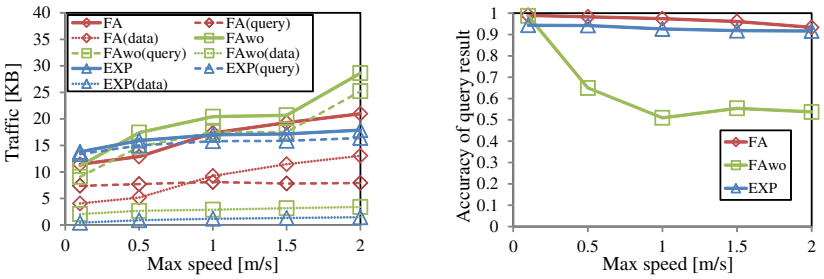
because of the movement of nodes when it issues a query. Therefore, it often happens that replies do not cover the Filling Area and some necessary data items are lacking. In the EXP method, the accuracy of query result is also lower than the FA method because all nodes within the search area reply to a query, which causes the collision of messages.

4.3 Impact of Requested Number of Data Items, k

We compare the FA method with the EXP and FAwo methods when varying k. Fig. 6 shows the simulation results. In the graphs, the horizontal axis indicates the requested number of data items, k, and the vertical axes respectively indicate the traffic in Fig. 6(a) and the accuracy of query result in Fig. 6(b).

From Fig. 6(a), as k increases, the traffic increases in all methods. This is because the search area for processing a kNN query and the number of data items in replies increase. In the FA method, the traffic for query processing becomes very small (almost zero) when k is smaller than C. This is because the query-issuing node can acquire all kNNs from its own cache. When k is small, the traffic for data maintenance in the FA method is relatively large when compared with that for query processing. Therefore, caching and maintaining data items are efficient when queries search in a wide area (e.g., k is large). On the other hand, the traffic in the FA method without maintenance becomes much larger as k increases. This is because as the search area increases, it happens more often that same data items are replied from multiple nodes.

From Fig. 6(b), in the FA method, the accuracy of query result is high, which is similar to the result in Fig. 5(b). In the FA method without maintenance, the accuracy of query result increases as k increases, because the search area becomes larger, and thus nodes within the search area incidentally retain kNNs more often. On the other hand, the accuracy of query result in the EXP method decreases as k increases, because packet losses often occur due to large number of replies. When k is 1, the accuracy of query result in the EXP method is low, because the search area is set very small, and thus it sometimes happens that no data items whose data point is within the search area are present.



(a) Traffic (b) Accuracy of query result

Fig. 7. Impact of Node Mobility

4.4 Impact of Node Mobility

Finally, we examine the performance of the FA method by varying node speed. Fig. 7 shows the simulation results. In the graphs, the horizontal axis indicates the maximum speed of nodes, v, and the vertical axes respectively indicate the traffic in Fig. 7(a) and the accuracy of query result in Fig. 7(b).

From Fig. 7(a), in all methods, as v increases, the traffic for data maintenance increases because the procedures of data maintenance occur more often. The traffic for query processing in the FA and EXP methods are almost constant, but that in the FA method without maintenance increases as v increases. This is because nodes move far from their update point, and thus, the mechanism for avoiding multiple replies of same items in the FA method does not work well.

From Fig. 7(b), in the FA and EXP methods, as v increases, the accuracy of query result slightly decreases because link disconnections occur more often. In the FA without maintenance, as v increases, the accuracy of the query result significantly decreases. This is because many nodes move far from their update point, the Filling Area cannot be covered by cached data items of nodes near the query point. Therefore, we confirmed that the data maintenance is important to keep high accuracy of query result when nodes move fast.

5 Conclusion

In this paper, we have proposed a kNN query method; the Filling Area (FA) method for searching location-dependent data items, which aim at reducing traffic and maintaining high accuracy of the query result in MANETs. In the FA method, to achieve a small search area, data items remain at nodes near the locations with which the items are associated, and nodes cache data items whose locations are near their own. When a node issues a query, neighboring nodes send back their copies, which will likely include the query result.

The experimental results show that the FA method reduces traffic for processing kNN queries and also achieve high accuracy of the query result. However,

the performance of the FA method degrades when the node mobility is high. As part of future work, we plan to extend our method to adapt to highly dynamic MANETs.

Acknowledgement. This research is partially supported by the Grant-in-Aid for Scientific Research (S)(21220002) and (B)(24300037) of MEXT, Japan.

References

1. Camp, T., Boleng, J., Davies, V.: A Survey of Mobility Models for Ad Hoc Networks Research. In: Wireless Communications and Mobile Computing (WCMC 2002), vol. 2(5), pp. 483–502 (2002)
2. Chatzimilioudis, G., Zeinalipour-Yazti, D., Lee, W.-C., Dikaiakos, M.D.: Continuous All k-Nearest Neighbor Querying in Smartphone Networks. In: Proc. of MDM 2012, pp. 79–88 (2012)
3. Hamida, E.B., Chelius, G.: A Line-Based Data Dissemination Protocol for Wireless Sensor Networks with Mobile Sink. In: Proc. of ICC 2008, pp. 2201–2205 (2008)
4. Heissenbüttel, M., Brtnoulli, T., Walchli, M.: BLR: Beacon-Less Routing Algorithm for Mobile Ad-Hoc Networks. Computer Communications 27(11), 1076–1086 (2004)
5. Komai, Y., Sasaki, Y., Hara, T., Nishio, S.: A kNN Query Processing Method in Mobile Ad Hoc Networks. In: Proc. of MDM 2011, pp. 287–288 (2011)
6. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
7. Nghiem, T.P., Waluyo, A.B., Taniar, D.: A Pure Peer-to-Peer Approach for kNN Query Processing in Mobile Ad Hoc Networks. In: Personal and Ubiquitous Computing (2012)
8. Padmanabhan, P., Gruenwald, L., Vallur, A., Atiquzzaman, M.: A Survery of Data Replication Techniques for Mobile Ad Hoc Network Databases. VLDB Journal 17(5), 1143–1164 (2008)
9. Scalable Network Technologies: "Qualnet," `http://www.scalable-networks.com/`
10. Song, Z., Roussopoulos, N.: K-Nearest Neighbor Search for Moving Query Point. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 79–96. Springer, Heidelberg (2001)
11. Tsuchida, G., Ishihara, S.: Replica Arrangement for Location Dependent Data in Consideration of Network Partition in Ad Hoc Networks. International Journal of Communication Networks and Distributed Systems 2(4), 401–423 (2009)
12. Yao, Y., Tang, X., Lim, E.-P.: Continuous monitoring of kNN queries in wireless sensor networks. In: Cao, J., Stojmenovic, I., Jia, X., Das, S.K. (eds.) MSN 2006. LNCS, vol. 4325, pp. 662–673. Springer, Heidelberg (2006)
13. Yashiro, T., Porta, T.F.L.: Nomadic Agent System: Infrastructureless location based service system. IPSJ Journal 46(12), 2952–2962 (2005)

Continuous Predictive Line Queries under Road-Network Constraints

Lasanthi Heendaliya, Dan Lin, and Ali Hurson

Department of Computer Science
Missouri University of Science and Technology
Rolla, MO, USA
{lnhmwc,lindan,hurson}@mst.edu

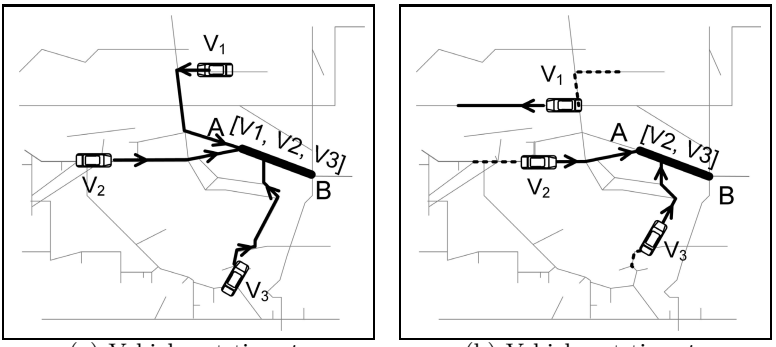
Abstract. With massively available global positioning systems, one can be up to date with its own position even when they are mobile. These position information, collectively, allows serving more knowledge to people on their neighborhood. This paper presents continuously monitoring predictive line query, which provides predicted future traffic information. The information would encourage the user to align his/her journey better, depending on the predicted traffic condition. Naturally, the accuracy of a prediction may become invalidated over time. The proposed query algorithm, thus, considers the continuous monitoring line query which keeps the issuer up-to-date over the time. If there is any significant change in the prediction results on the querying road due to location updates of other vehicles, the updated query result will be automatically sent back to the user. To speed up query processing, a novel data structure is designed, the TPR^Q-tree. The TPR^Q-tree facilitates one update message to be considered on a group of queries. This group wise consideration, contrary to the individual consideration, has reduces the execution time significantly. The results of extensive experimental study demonstrate the efficiency and effectiveness of proposed approach.

1 Introduction

Nowadays more and more vehicles are equipped with Global Positioning Systems (GPS) and navigation systems, and more and more users have mobile devices like smart phones equipped with GPS. The new technology has helped greatly enhance driving experience. For example, finding an optimal route and checking the real-time traffic condition has become a common practice for many drivers. In this work, we aim to further advance the existing technology on traffic monitoring and incorporate the spirit of ubiquitous computing to provide even better experience for users.

In particular, we observe that most existing traffic monitoring applications only provide *current traffic condition*. However, the route calculation based on *current traffic condition* may not be optimal. Consider the following example. Bob plans to travel from Rolla to St. Louis which is about 100 miles (i.e., about 2-hour driving). When he sets off, there is a traffic jam on his way to St. Louis.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 228–242, 2013.



(a) Vehicles at time t_1 (b) Vehicles at time t_2

Fig. 1. Dynamacity of Continuous Traffic Prediction

If the navigation system computes the travel route for Bob based on current traffic condition, the route will probably include a detour to bypass the traffic jam. However, an hour later when Bob is already on his detour route, the traffic jam has been cleared up. Bob actually needs not take the detour if the navigation system is able to calculate the route with predicted traffic condition. This kind of scenarios inspire us to design a traffic prediction system that can provide better insight in travel planning. Moreover, the traffic prediction should be proactive/pervasive in that once the user initiates a traffic condition prediction query, the system continuously monitors the prediction results and reports any changes that may be caused by the dynamic traffic. Figure 1 illustrates an example of continuous traffic prediction.

Figures 1(a) and 1(b) show snapshots of three vehicles at time t_1 and t_2 respectively. The query road segment is $\overline{AB}$, and the current travel plans of the vehicles are highlighted by bold lines. As shown in Figure 1(a), three vehicles V_1, V_2, and V_3 may enter the querying road $\overline{AB}$. However, as time passes, vehicle V_1 changes its travel plan by making a right turn earlier at time t_2. As a result, only two vehicles (V_1 and V_3) may enter the query road which requires an update of the previous query results.

To build the above envisioned system, none of the existing approaches can be directly adopted. The closely related work that can provide traffic information includes range queries and density queries. A range query reports traffic information in a given circular or rectangular area [2, 7, 20], which contains traffic information on irrelevant roads rather than just the routes that the query issuer may pass by. The density query [8, 12, 17, 21] outputs even coarse information which are regions with vehicles more than certain threshold. More importantly, these range queries and density queries are mostly designed for objects moving in Euclidean space without road network constraints. Therefore, they are not able to provide precise traffic prediction. Very few works [4,9] can be found that consider road network constraints. Those few, however, only support queries on current traffic condition but not traffic prediction.

In this paper, we propose a solution to the construction of the continuous traffic prediction system. We formalize the problem as a new type query, namely *continuous predictive line query*. The continuous predictive line query allows a user to specify a road that he/she would like to know about the traffic condition of. Then, the query returns predicted traffic condition of the querying road at the estimated time that the user may pass by. If there is any significant change of the prediction results on the querying road due to location updates of other vehicles, the updated query result will be automatically sent back to the user. To speed up query processing, we design a novel data structure, the TPR^Q-tree, which indexes queries and efficiently handles the query result update that evolve with time. We have conducted an extensive experimental study and the results demonstrate the efficiency and effectiveness of our approach.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 provides the problem statement. Section 4 introduces the proposed index structure. Section 5 presents the query algorithms. Experimental results are presented in Section 6 and finally, Section 7 concludes the paper.

2 Related Work

By definition, the term continuous refers to the continuous monitoring of an issued query. This continuity, however, can come in two forms: continuous monitoring of static query or continuous monitoring of dynamic queries. In the former category, query specifications do not change over the time, but the query result might be changed due to other moving objects. Thus, the query result is continuously monitored. In the later category, the query specifications can also change over time as the query issuer moves. In either case, frequent and the large amounts of update messages should be handled in an efficient manner for up-to-date query results. Among the existing location dependent query types, the closest related query types which can provide real time traffic information are *range queries* and the *density queries*. However, these query types cannot provide future traffic information [10,24]. In addition to the those two query types, other query types have been also proposed for continuous static query monitoring. Examples include KNN, top-K queries [15], Reverse KNN [22], Detour queries [18].

A common approach to continuous monitoring on static queries is to define a safe region for moving objects [2,7,13,14,19], as any movement of objects within the safe region does not alter the result. These safe regions could be maintained by either the query contributors [2, 13, 19] or the query issuer [14]. The safe regions proposed in [2] has less computation cost than that of in [19]. Additionally, [19] provides adjustable safe regions depending on the computational capability of objects as well. However, it is not clear what motivates a mobile node to monitor a bigger range w.r.t to other nodes to get the same service. Additionally, regardless of who maintains the safe regions, there is a possible privacy violation of individual objects. The influence region maintained in the proposed algorithm, does not violate anyone's privacy. At the same time, the computation cost and the maintenance cost of the region are also less. That is because,

the influence regions are defined by two circles, which are easy to maintain and only the radii will be changed with the time. In contrary to the aforementioned technique, the algorithm proposed by Wang and Zimmermann [20], first get the snapshot query result and maintain that initial result. The main drawback of this approach is the amount of data managed in the memory as only the road network is stored in the secondary disk in a R-tree structure. Thus, memory issue consequently becomes severe when the number of queries in one cell increase. Proposed approach also follows a similar technique, but it is more scalable as it is disk based.

Most approaches proposed for the static query do not fit well for dynamic queries without modifications. For instance, the safe region calculated for the static queries may not be valid when the query is also moving. Also, the safe region is more dynamic for moving queries compared to the static queries.

Stojanovic et al. [3] proposed algorithms for processing dynamic continuous range queries, in which there are no special pruning techniques applied to prune unnecessary queries and update messages consideration. As a consequence, their approach has a higher computational cost. Algorithms proposed by Liu and Hua [11] process dynamic range and KNN queries, which considers the network distance. This approach shows better performance when the range is bigger since the number of messages to be communicated with the server is less. However, similar to that of [3], this approach also maintains a considerable amount of data on memory, including some duplicated information.

Gedik and Liu [5] proposed an distributed approach for spatial queries to reduce the work load at the server. This approach distributes the responsibility of monitoring query result changes to other vehicles. Although, this reduces the communication cost, it violates the confidentiality of query issuers. Both SEA-CNN: Shared Execution Algorithm [23] and SCUBA: Scalable Cluster-Based algorithm [16] try to provide scalability in spatio-temporal queries by considering the group wise execution. These methods are better if the steady groups are exhibited. Otherwise, the number of clusters to be considered becomes high which increases the query update cost.

3 Problem Statement

In this section, we present the formal definition of the *continuous predictive line query*. The continuous predictive line query is developed based on the predictive line query as introduced in [6].

Definition 1. *[Predictive Line Query] A predictive line query* $PLQ = (e_q, t_q, t_c)$ *retrieves all moving objects which will be on the query road segment* e_q *at the query time* t_q*, where* $t_q > t_c$ *and* t_c *is query issuing time.*

The predictive line query is a one-time snapshot query. It does not consider possible changes of the predicted traffic condition when the query issuer moves closer to the querying road. In order to provide timely and up-to-date information to the query issuer, we define a continuous version of the predictive line query as follows.

Definition 2. *[Continuous Predictive Line Query] A continuous predictive line query* $CPLQ = (e_q, t_q, t_c, \rho)$ *continuously monitors the moving objects which will be on the query road segment* e_q *at the query time* t_q*, and returns query results whenever the number of query results differ more than a threshold* ρ*. Specifically, let* R_i *denote the query results at time* t_i $(t_c \leq t_i \leq t_q)$*, CPLQ returns the query results in the form of* $\{(R_1, t_c), (R_2, t_2), ..., (R_k, t_q)\}$*, and* $|R_{i+1}| - |R_i| > \rho$.

For example, a CPL query like $CPLQ$=($\overline{AB}$, 8am, 7:30am, 20) means that the query issuer issued a query at 7:30am and is interested in the traffic at road $\overline{AB}$ at 8am. The query issuer expect the server to report the change of prediction results if the difference of the number of vehicles on the querying road is more than 20. Note that it is not necessary for the query issuer to specify the threshold parameter. Instead the threshold can be automatically chosen by the server according to the past experience to reflect significant traffic change.

4 Data Structures

In this section, we introduce the two major data structures employed to support the continuous predictive line query. One is the R^D-tree [6] for management of both the road networks and moving object information. The other is our proposed Time-Parameterized R*-tree for Query (called TPR^Q-tree), which indexes query information. In what follows, we describe the two data structures in detail.

4.1 The R^D-Tree

We adopt the R^D-tree since it is the most recent index that supports snapshot predictive line queries on moving objects under the road network constraints. The R^D-tree is composed of an R*-tree and a set of hash tables. The R*-tree indexes the road network data. Hash tables store the moving object information.

The road network is represented as a graph $G(E, V)$; where E is the set of edges, and V is the set of vertices. Each edge $e \in E$ represents a road segment[1] in the network. Here, $e = \{v_1, v_2\}$, where $v_1, v_2 \in V$; v_1 and v_2 are starting and end nodes of the road segment respectively. Furthermore, each edge is associated with two parameters: l and s, where l is the length of the edge and s is the maximum possible speed on that edge. Each edge also maintains a list of vehicles moving on itself.

A moving object (vehicle) O is represented by the tuple $\{vId,\ x_1, y_1, e_c,\ e_n,\ speed,\ e_d,\ t\}$, where vId is the unique ID of the vehicle, x_1 and y_1 are the coordinates of the vehicle at the latest update timestamp t, e_c is the current road segment that the vehicle is on, e_n is the next road segment that the vehicle is heading to, and e_n is the vehicle's traveling destination. Here, it is assumed that most moving objects are willing to disclose their tentative traveling destinations to the service provider (server) in order to obtain high-quality services. However

[1] Road segment and edge may be used interchangeably throughout this paper.

the destination may change during the trip. Moving objects are grouped according to the geographical direction formed according to individual's destination with respect to the current position.

4.2 The TPR^Q-Tree

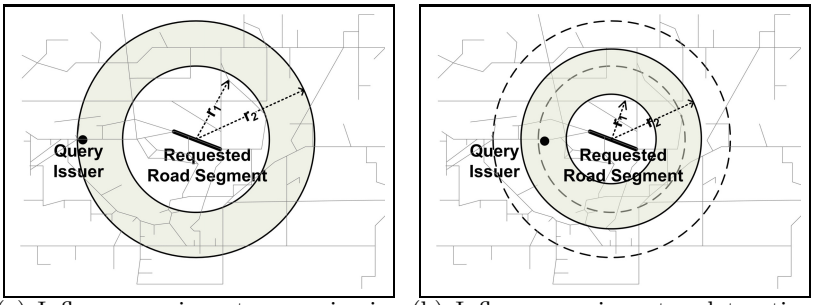
The CPL queries require to continuously report the moving objects on the query road segment at a near future timestamp. A naive approach for answering a continuous query is to re-conduct the same query every timestamp till the expiration of the life time of the continuous query. This may involve lots of unnecessary efforts if there is no change of the query results at consecutive timestamps. Observing more closely, there is a need to update the query results only when an object in the current result becomes invalid or a new object joins the result due to the change of their moving functions. Given a large amount of moving object updates occurring every timestamp, we propose TPR^Q-tree to be used to facilitate quick identification of which update affects which CPL query in order to achieve efficient query performance.

The TPR^Q-tree does not simply index the query road segment of a CPL query. Instead, the TPR^Q indexes an *influence region* for each CPL query. The *influence region(IR)* is the region which covers majority of moving objects that may enter the query road segment at the future query time. In other words, if objects in this IR update their movement functions, the query results may be affected.

To have a better understanding of the IR, let us consider the example shown in Figure 2. Query Q_1 aims to predict moving objects entering the highlighted road in 30 minutes from now. Figure 2(a) shows the IR of Q_1 at query issuing time. From the figure, we can see that the IR has a ring shape. Its inner radius is the road distance that can be traveled in 30 minutes (the query interval) by an object. On the contrary, its outer radius is the road distance that can be traveled by an object with fastest moving speed in 30 minutes. All moving objects covered by this ring have the possibility to enter the query road segment. The interesting issue is that as time evolves, the IR will shrink as shown in Figure 2(b). This is because the time to travel to the query road segment is shortened as the time getting closer to the future query time. More specifically, at the query issuing time, the CPL query considers objects which travel 30 minutes to enter the road segments. After 10 minutes of the query issuing time, the CPL query considers objects which takes 20 minutes to enter the road segments.

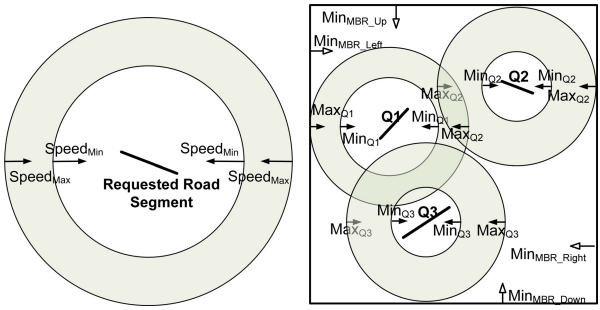
To model the shrinking IR,it is stored as the parameterized ring which has moving speed attached to both inner and outer radius. The inner radius is associated with a minimum moving speed towards the query road segment while the outer radius is associated with a maximum speed towards the query road segments as shown by the arrows in Figure 3(a). The time-parameterized IR is formally defined as follows.

Definition 3. *[Influence Region] Let $Q = (e_q, t_q, t_c, \rho)$ be a CPL query. The influence region is a time-parameterized ring in the form of $IR = (c, r_1, v_1, r_2, v_2)$,*



(a) Influence region at query issuing time: t_c (b) Influence region at a later time stamp: t'_c

Fig. 2. Shrinking Influence Region at Time t_c and $t'_c(> t_c)$



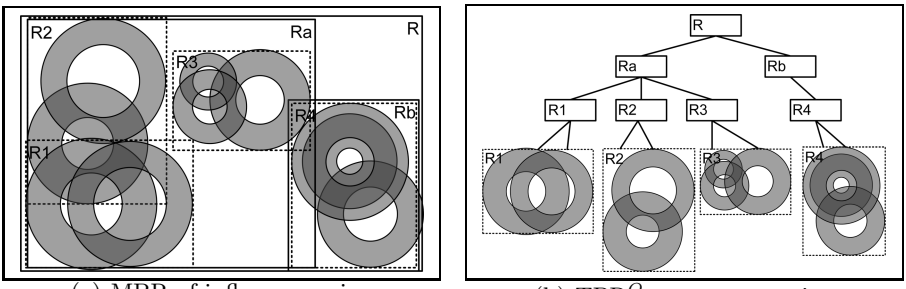
(a) Speed of influence region's radii (b) Speed of MBRs

Fig. 3. Speeds of Influence Region and MBR

where c is the middle point of querying road e_q, r_1=*RoadDist*$(v_{min} \cdot (t_q - t_c))$ *and* r_2=*RoadDist*$(v_{max} \cdot (t_q - t_c))$. *The* v_1 *and* v_2 *are the speed of the inner and the outer circles respectively.*

We now proceed to introduce the structure of the TPR^Q-tree. Figure 4(a) illustrates the structure of a TPR^Q-tree. The base structure of the TPR^Q-tree is the R*-tree. There are three types of nodes in the TPR^Q-tree: leaf nodes; immediate parent node of the leaf nodes; higher-level internal nodes. We elaborate their structure as follows:

- Leaf level: An entry in the leaf node of the TPR^Q-tree stores information of a CPL query which includes: the query parameters (e_q, t_q, t_c, ρ), the corresponding influence region IR, a list of query issuers, and a pointer to the query results.
- Second level: Each entry in the parent node of the leaf nodes stores a pointer to the leaf node and a time-parameterized minimum bounding rectangle (MBR) that bounds all the IRs of the queries in the leaf node. The



(a) MBR of influence region (b) TPR^Q-tree construction

Fig. 4. An TPR^Q-tree Construction

time-parameterized MBR has a speed attached to each edge as shown in Figure 3(b). The speed of each edge is the minimum speed among the speeds of outer rings of all IRs in the MBR. The moving direction of each edge is pointing to the center of the MBR so that the MBR shrinks as time passes and bounds the shrinking IRs. The time-parameterized MBR is stored as a six-tuple $(x_1, y_1, x_2, y_2, v, t_u)$ where (x_1, y_1) is the coordinates of the left lower corner of the MBR, (x_2, y_2) is the right upper corner of the MBR, v is the speed of each edge and t_u is the latest time that the parameters of the MBR is updated.

– Higher levels: An entry in higher level internal nodes contain a pointer to the child node and a time-parameterized MBR that bounds MBRs in the child node. Each edge of the MBR is associated with a minimum speed among the speeds of its child MBRs and each edge is moving towards the center as well.

Construction and Maintenance of the TPR^Q-Tree. There are three basic operations in the TPR^Q-tree: inserting a new query, deleting an existing query, and updating an existing query.

Given a new CPL query, we first compute its IR (denoted as IR_{new}). Then, we start the search from the root of the TPR^Q-tree to find the proper leaf node to store this new query. At each level of the TPR^Q-tree, we compute the MBRs at current time based on their shrinking speed. We first consider the MBR that can fully cover IR_{new}. If such MBR does not exist, we consider the MBR with the minimum enlargement to include IR_{new}. Finally, the CPL query is inserted to the leaf node. The speed and the size of the MBR in the parent entry of the leaf node may need to be updated accordingly. The update may propagate all the way up to the root node. In addition, if the node is full, first some entries from the full node is removed from the tree and reinserted, to seek any appropriate nodes within the existing structure. If a full node is found in this reinsertion, a node split may occur. For the node split, first the axis (x or y) on which the split is going to performed is selected. Then the entries in the node are sorted according to the coordinate of the selected axis. The split index from the array is selected as the index which divides objects into two with least MBR areas.

A CPL query needs to be deleted from the TPR^Q-tree either when the query issuer passed the querying road segment or when the issuer withdraws the query by him/herself before the query expires. To delete the query, we search from the root of the TPR^Q-tree and checks all the MBRs that fully covers the query's IR until identify the specific leaf node that contains the query to be deleted. Once the query is deleted, the MBR in the parent entry of this leaf node may need to be updated as well. In addition, if the deletion causes under-flow, node merging will occur.

A query update involves two phases. The first phase is to locate the query in the TPR^Q-tree. Then, we check if the query with the new parameters is still covered in the same leaf node. If so, we just need to update the query parameters, and the MBR of this leaf node if the new query parameter changes the MBR's speed. If the new query can no longer be included in the current leaf node, we will conduct a deletion and then treat the new query as a new insertion.

5 Continuous Predictive Line Query Algorithms

The CPL query algorithm consists of two phases: the initial phase and the maintaining phase. The initial phase computes the query result for a newly issued query according to the current moving object information. The maintaining phase, monitors the query result obtained at the initial phase and adjusts them upon updates of moving objects, until the query expires.

5.1 Initial Phase

Upon receiving a new query from a user u, the TPR^Q-tree is first searched to check if the new query has the same parameter as an existing query. Two queries are considered the same if they are querying traffic of the same road segment at the same near future timestamp. If the same query can be found in the tree, the user u is added to the query issuer ID list and directly report the query results. In practice, it can be expected that many people might be interested on some particular road segments. That could be because the road segment has often distinguished for traffic congestions, or it could be a hub for many popular destinations. Thus, in this kind of situation, searching the query in the tree would save the repeated query execution cost.

If the new query does not exist in the TPR^Q-tree, the following two steps will be conducted. First, insert the new query to the TPR^Q-tree. Note that the insertion can take advantage of the previous search. In previous search, we already reach the leaf node that may contain the same query as the new query. From another point of view, this leaf node is the most appropriate place to insert the new query.

The second step is to execute a snapshot predictive line query based on current moving object moving functions. Specifically, the R^D-tree is searched to find the road segments covered by the IR of the query. Then, the objects in the retrieved road segments are checked. Depending on the geographical direction of

the query road segment with respect to each edge, the hash bucket with the same direction is accessed to find objects currently moving on it. This set of candidate objects are then filtered in the second phase. This phase considers the individual candidate's tentative path, the shortest route, to decide the possibility to enter the query road at the query time.

5.2 Maintaining Phase

The prediction results obtained from the initial phase may need to be updated upon changes of some vehicles' travel plans just as shown in Figure 1 in the introduction.

If a vehicle changes its moving direction or speed dramatically, the vehicle will send an update to the server. Upon receiving the update message, the server will check if the update affects existing queries by answering the following two questions: (1) Is this object currently included in any existing query result? (2) Is this object going to be in some queries results' after the update? Given an object update and one query, there are four cases for the above two questions:

1. The object is included in the query result, and is still the query result after the update.
2. The object is included in the query result but will no longer be valid query result after the update.
3. The object is not included in the query result but will become the query result after the update.
4. The object is not included in the query result and will also not be the query result after the update.

The challenge is how to efficiently categorize each update message into one of the four cases against all existing CPL queries. To achieve this, we leverage the proposed TPR^Q-tree as follows.

We aim to utilize the TPR^Q-tree to identify two sets of queries for each update message: (i) Q_{old}: queries to which the object belongs to at the object's previous update timestamp t_{old}; (ii) Q_{new} queries to which the object belongs to after the update at current timestamp t_c. These two sets of queries are identified simultaneously in one round of search in the TPR^Q-tree. In particular, we start the search from the root of the tree. For each entry of the visited node, we compute its IRs at t_{old} and t_c respectively. The center of the IR is the same while the inner and outer radii are computed based on the inner/outer speed multiplied by the time difference as follows:

$$r_{old\_inner} = RoadDist(v_{min} \cdot (t_q - t_{old})), r_{old\_outer} = RoadDist(v_{max} \cdot (t_q - t_{old}))$$

$$r_{new\_inner} = RoadDist(v_{min} \cdot (t_q - t_c)), r_{new\_outer} = RoadDist(v_{max} \cdot (t_q - t_c))$$

If the object's previous position is included either in the old or new IR, we keep searching the children nodes of this entry until reach the leaf node. Given an entry in the leaf node, if the object's position is included in the old IR of the

query, the query is inserted to Q_{old}; if the object's updated position is included in the new IR of the query, then the query is inserted to Q_{new}.

Next, we analyze the collected sets of Q_{old} and Q_{new}. If the same query ID appears in both Q_{old} and Q_{new}, no change to the query result is needed, which is the first case. If a query ID appears only in Q_{old} but not Q_{new}, that means case 2 and we remove the object from the query result. If a query ID appears only in Q_{new} but not Q_{old}, that means case 3 and we insert the object to the query result. If both sets are empty, that means the object does not affect the previous and current query results, which is case 4.

Finally, we record the number of changes for each query result during the object update. If the number exceeds the specified threshold ρ, the server will return the latest query results to the query issuer.

6 Experimental Results

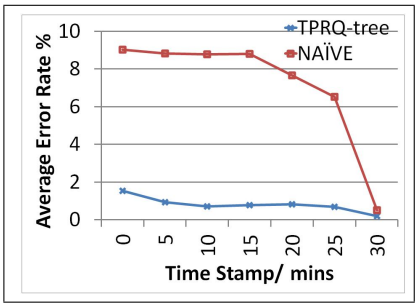
We use the Brinkhoff's generator [1] to generate moving objects on real road maps. The number of moving objects in each dataset ranges from 10K to 100K. The object speeds range from 30mph to 60mph. Four maps, with different topology but similar number of road segments were selected as inputs to the generator. The Worth county in Missouri state was selected as the default road map. We generate continuous predictive line queries by randomly selecting query road segment and predictive time length. Table 1 summarized the experimental parameters and the default values are highlighted in bold.

Table 1. Simulation Parameters and Their Values

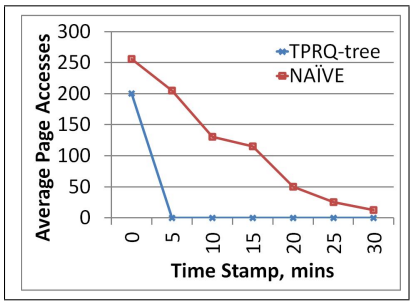
Parameters	Values
number of moving objects	10K, 20K, ..., **50K**, 60K, ..., 100K
predictive time length	10, 20, **30**, 40, 50, 60 (minutes)
road maps	**Worth(MO)**, Alpine (CA), Charles (MD), Salem (NJ)

We compare our approach with a naive approach that executes snapshot predictive line queries [6] every timestamp. The performance is measured in terms of I/O cost (the number of disk-page accesses) and prediction error rate. When measuring the page accesses, we assumed a 50k buffer. The error rate is computed by comparing the number of objects in the predictive query results with the actual number of objects on the query road segment at the query time. Each test case was run for 250 queries and the average cost is reported.

Behavior over the Query Life Time. First, we present the behavior of performance on discrete time stamps during the query life time. Given 30 minutes prediction time interval, we record the error rate and the page accesses as the current time gets closer to the query time. Figure 5(a) and 5(b) reports the performance of our approach (denoted as TPR^Q-tree) and the naive approach.

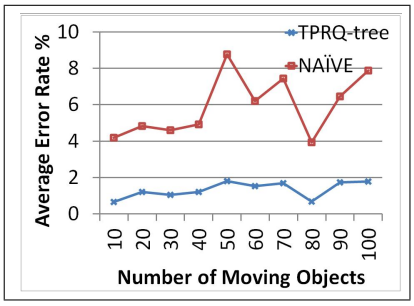


(a) Error Rate

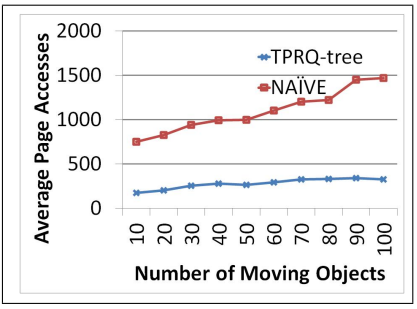


(b) Page Accesses

Fig. 5. Behavior of the Query Life Time



(a) Error Rate



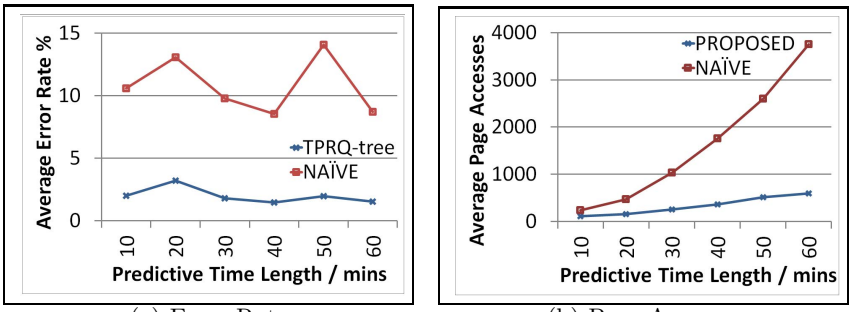
(b) Page Accesses

Fig. 6. Effect of Number of Moving Objects

As shown in Figure 5(a), we can observe that the TPR^Q-tree has much less error rate than the naive approach. This is because the naive approach adopts a ring query which is defined based on Euclidean distance to the query road segment [6], whereas the influence region employed by the TPR^Q-tree considers the road distance and hence it is more accurate to enclose vehicles that may enter the query road segment.

From Figure 5(b), we can see that our approach is much more efficient than the naive approach. The reason is the following. The naive approach needs to execute each query every timestamp which may involve duplicate efforts when there is no change to the results. The TPR^Q-tree takes on object update and then check all affected queries simultaneously, which has helped significantly reduced unnecessary efforts on query processing.

Effect of Number of Moving Objects. In this round of experiments, we evaluate the performance when the number of moving objects increases from 10K to 100K. Figure 6(a) shows the average error rate of both approaches. The error rates in both approaches increase slightly with the number of objects. This is because the more moving objects, the more uncertainty of the prediction.



(a) Error Rate (b) Page Accesses

Fig. 7. Effect of Predictive Time Length

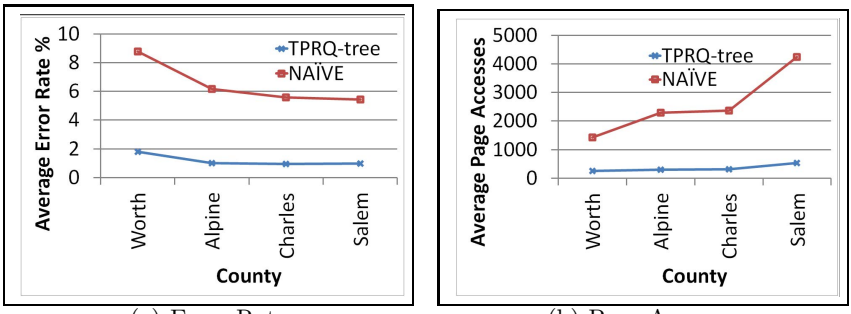
However, our approach always achieves a lower error rate for the same reason discussed in the previous section.

Regarding the page accesses, Figure 6(b) shows that the naive approach increases a little faster than our approach. The possible reason is that the naive approach needs to execute the query on a larger number of objects and hence requires more page accesses. The TPR^Q-tree indexes queries rather than objects. Given an object update, the cost to revise the query results is similar. The small increase is mainly due to the increase of object updates per timestamp with the increase of the dataset size.

Effect of Predictive Time Length. In this set of experiments, we vary the predictive time length from 10 minutes to 60 minutes. As shown in Figure 7(a), the error rate stays in a similar range regardless of the predictive time length for both approaches. The behavior can be explained as follows. For the naive approach, it executes the query every timetamp and hence any change of object travel plan will be captured. Similarly in our approach, we consider the effect of the object update on the query results at every timestamp.

On the other hand, the predictive time length does affect the query cost as shown in Figure 7(b). The query cost of both approaches increases when the predictive time length is longer. This is because in the naive approach, a bigger ring query is generated given a longer predictive time length. In our approach, a bigger influence region is generated based on the formula given in Definition 3, which results in examining more candidate objects. However, our approach always performs better than the naive approach which is again attributed to the TPR^Q-tree.

Effect of Road Topology. Finally, we evaluate the effect of the road topology by testing different maps: Worth (MO), Alpine (CA), Charles (MD), and Salem (NJ). The number of edges in each map was 1573, 1576, 1766, and 1789 respectively, and the average road segment length is 551m, 232m, 370m, and 515m, respectively. By observing the average error rate of individual topology in Figure 8(a), it is tend to conclude that the larger the number of edges, the lower the



(a) Error Rate (b) Page Accesses

Fig. 8. Effect of Road Topology

error rate. Regarding the page accesses as shown in Figure 8(b), our approach is relatively independent of the road topology, while the naive approach yields higher cost with the growth of the number of edges.

7 Conclusion and Future Work

In this paper, we define a new type of query, namely continuous predictive line query, which takes into account road network constraints in predicting traffic on a given road segment. To answer the query, we propose the TPR^Q-tree and an efficient query algorithm. The TPR^Q-tree is a time parameterized tree that indexes queries' influence regions that shrink as time evolves. Our experimental study has demonstrated the efficiency and effectiveness of our approach.

As future work, we will seek real traffic data to further verify our approach.

Acknowledgement. This work is partly funded by the U.S. National Science Foundation under Grant No. CNS-1250327.

References

1. Brinkhoff, T.: A framework for generating network-based moving objects (2004)
2. Cai, Y., Hua, K.A., Cao, G.: Processing range-monitoring queries on heterogeneous mobile objects. In: Proceedings of the 2004 IEEE International Conference on Mobile Data Management (2004)
3. Predic, B., Papadopoulos, A.N., Stojanovic, D., Djordjevic-Kajan, S., Nanopoulos, A.: Continuous Range Query Processing for Network Constrained Mobile Objects. In: 8th International Conference on Enterprise Information Systems (2006)
4. Feng, J., Lu, J., Zhu, Y., Mukai, N., Watanabe, T.: Indexing of moving objects on road network using composite structure. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part II. LNCS (LNAI), vol. 4693, pp. 1097–1104. Springer, Heidelberg (2007)
5. Gedik, B., Liu, L.: Mobieyes: A distributed location monitoring service using moving location queries. IEEE Transactions on Mobile Computing (2006)
6. Heendaliya, L., Lin, D., Hurson, A.R.: Predictive Line Queries for Traffic Forecasting. In: Database and Expert Systems Applications (2012)

7. Hu, H., Xu, J., Lee, D.L.: A generic framework for monitoring continuous spatial queries over moving objects. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2005)
8. Jensen, C.S., Lin, D., Beng, C.O., Zhang, R.: Effective density queries on continuously moving objects. In: Proceedings of the 22nd International Conference on Data Engineering (2006)
9. Kyoung-Sook, K., Si-Wan, K., Tae-Wan, K., Ki-Joune, L.: Fast Indexing and Updating Method for Moving Objects on Road Networks. In: Proceedings. 4th International Conference on WISEWs (2003)
10. Kang, H.-Y., Kim, J.-S., Li, K.-J.: Indexing moving objects on road networks in p2p and broadcasting environments. In: W2GIS (2006)
11. Liu, F., Hua, K.A.: Moving query monitoring in spatial network environments. Mob. Netw. Appl. (2012)
12. Gunopulos, D., Hadjieleftheriou, M., Kollios, G., Tsotras, V.J.: On-line discovery of dense areas in spatio-temporal databases. In: International Symposium on Advances in Spatial and Temporal Databases, SSTDn (2003)
13. Mouratidis, K., Papadias, D., Bakiras, S., Tao, Y.: A threshold-based algorithm for continuous monitoring of k nearest neighbors. IEEE Trans. on Knowl. and Data Eng (2005)
14. Mouratidis, K., Papadias, D., Hadjieleftheriou, M.: Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2005)
15. Mouratidis, K., Yiu, M.L.G., Papadias, D., Mamoulis, N.: Continuous Nearest Neighbor Monitoring in Road Networks. In: Proceedings of the 32nd International Conference on Very Large Data Bases (2006)
16. Nehme, R.V., Rundensteiner, E.A.: SCUBA: Scalable cluster-based algorithm for evaluating continuous spatio-temporal queries on moving objects. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 1001–1019. Springer, Heidelberg (2006)
17. Ni, J., Ravishankar, C.V.: Pointwise-dense region queries in spatio-temporal databases. In: IEEE 23rd International Conference on Data Engineering (2007)
18. Nutanong, S., Tanin, E., Shao, J., Zhang, R., Kotagiri, R.: Continuous detour queries in spatial networks. IEEE Transactions on Knowledge and Data Engineering (2012)
19. Prabhakar, S., Xia, Y., Kalashnikov, D.V., Aref, W.G., Hambrusch, S.E.: Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. IEEE Trans. Comput. (2002)
20. Wang, H., Zimmermann, R.: Processing of continuous location-based range queries on moving objects in road networks. IEEE Transactions on Knowledge and Data Engineering (2011)
21. Wen, J., Meng, X., Hao, X., Xu, J.: An efficient approach for continuous density queries. In: Frontiers of Computer Science (2012)
22. Xia, T., Zhang, D.: Continuous reverse nearest neighbor monitoring. In: Proceedings of the 22nd International Conference on Data Engineering (2006)
23. Xiong, X., Mokbel, M.F., Aref, W.G.: Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In: Proceedings of the 21st International Conference on Data Engineering (2005)
24. Yang, Y.C., Cheng, C.M., Lin, P.Y., Tsao, S.L.: A Real-Time Road Traffic Information System based on a Peer-to-Peer Approach. In: IEEE Symposium on Computers and Communications (2008)

Sensitivity Analysis of Answer Ordering from Probabilistic Databases

Jianwen Chen, Yiping Li, and Ling Feng

Dept. of Computer Science & Technology, Tsinghua University, Beijing, China
{chen-jw08,liyp09}@mails.tsinghua.edu.cn, fengling@tsinghua.edu.cn

Abstract. Queries over probabilistic databases result in probabilistic answers, which are often ranked according to certain ranking criteria. As the probabilities of the basic tuples may be imprecise and erroneous, and their perturbations may lead to great changes in answer ordering, sensitivity analysis like "*which basic input probability change can substantially alter the ranked result?*", "*which basic probability change will make a certain element top-ranked?*", "*which basic probability change will swap the positions of the firstly and secondly ranked elements?*" thus arise.

The sensitivity analysis of top-K probabilistic query has been touched in the literature, mainly concerning the change of the answer list as a set. However, the ordering of the elements in the answer list matters highly for certain applications. In this paper, we categorize a variety of such kinds of ordering sensitivity questions into *list*-oriented or *element*-oriented, and formulate the sensitivity analysis problem for answer ordering returned from probabilistic top-K queries and probabilistic top-K aggregation queries. We develop a modular approach to quantitatively compute sensitivity of answer ordering, where four basic processing modules are identified. Optimization strategies are also presented for performance improvement. Experimental results on both synthetic and real data demonstrate the effectiveness and efficiency of the proposed solutions.

Keywords: Probabilistic database, top-K query, aggregation query, answer ordering, sensitivity analysis.

1 Introduction

1.1 Motivation

Uncertain data is consistently emerging in a variety of application domains like sensor network, statistical data analysis, machine learning and data mining, information extraction and integration, etc. To manage data uncertainty, a community of efforts have centered around probabilistic data management [1,4]. In probabilistic databases, basic tuples are stored with probabilities reflecting their confidence. Derived result tuples based on the basic ones are also accompanied with probabilities accordingly. Thus, they are usually ranked according to certain ranking criteria, whose orderings are sensitive to perturbations of basic tuples' probabilities in the database.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 243–258, 2013.

Restaurant

	rid	*name*	*district*	*prob.*
e_1	1	KFC	West	0.9
e_2	2	McDonald	West	0.8
e_3	3	BurgerKing	East	0.7
e_4	4	PizzaHut	South	0.6
e_5	5	Subway	North	0.5
e_6	6	Arby's	North	0.4

ResReputation

	rid	*reputation*	*prob.*
e_7	1	Good	0.8
e_8	2	Good	0.7
e_9	3	Good	0.6
e_{10}	4	Good	0.5
e_{11}	5	Medium	0.4
e_{12}	6	Bad	0.3

ResDiscount

	rid	*discount*	*prob.*
e_{13}	1	8	0.7
e_{14}	2	7	0.6
e_{15}	3	6	0.5
e_{16}	4	5	0.4
e_{17}	5	9	0.3
e_{18}	6	9	0.2

Fig. 1. A probabilistic database example

Query:*find top-3 restaurants with good reputation and discount less than 9.*

```
SELECT name
FROM Restaurant, ResReputation, ResDiscount
WHERE Restaurant.rid = ResReputation.rid AND
Restaurant.rid = ResDiscound.rid AND
ResReputation.reputation = 'Good' AND
ResDiscount.discount < 9
ORDER BY prob. DESC LIMIT 3
```

Top-3 answer list

event expression	*name*	*Prob.*
$e_1 \wedge e_7 \wedge e_{13}$	KFC	0.504
$e_2 \wedge e_8 \wedge e_{14}$	McDonald	0.336
$e_3 \wedge e_9 \wedge e_{15}$	BurgerKing	0.21

Sensitivity Question 1: *which basic input probability change will make KFC drop out of top-3?*

event expression	*name*	*prob.*
$e_2 \wedge e_8 \wedge e_{14}$	McDonald	0.336
$e_3 \wedge e_9 \wedge e_{15}$	BurgerKing	0.21
$e_4 \wedge e_{10} \wedge e_{16}$	PizzaHut	0.12

Answer: $p(e_{13})$: $0.7 \rightarrow 0.167$

Sensitivity Question 2: *which basic input probability change will swap the positions of KFC and Mcdonald?*

event expression	*name*	*prob.*
$e_2 \wedge e_8 \wedge e_{14}$	Mcdonald	0.336
$e_1 \wedge e_7 \wedge e_{13}$	KFC	0.331
$e_4 \wedge e_{10} \wedge e_{16}$	BurgerKing	0.12

Answer: $p(e_{13})$: $0.7 \rightarrow 0.46$

Fig. 2. Two sensitivity questions on a probabilistic top-K query answer

Example 1. Consider an information extraction application, which extracts information about restaurants from the Web. A probabilistic database in Figure 1, including three example tables (`Restaurant`, `ResReputation`, and `ResDiscount`) is formed, showing restaurants' possible locations, reputations, and discounts. Each basic tuple represents a probabilistic basic event, and has an event identifier and a probability attribute. Figure 2 shows a top-3 answer list (ranked according to result probability) to the probabilistic query for the restaurants with good reputation and discount less than 9. Each answer tuple's probability is computed from the probabilities of its deriving basic tuples. For example, for the first tuple KFC, its probability is computed as $P(e_1 \wedge e_7 \wedge e_{13}) = 0.9 * 0.8 * 0.7 = 0.504$. Two sensitivity concerns regarding the top-3 answer list may arise. (1) which basic input probability change will make KFC to drop out of top-3? (2) which basic input probability will swap the positions of KFC and McDonald? When $p(e_{13})$ changes from 0.7 to 0.167, result tuple KFC will drop out of top-3. Also, if $p(e_{13})$ drops from 0.7 to 0.467, result tuple KFC and McDonald will swap their first-second positions. □

As illustrated, the answer ordering returned from a probabilistic database query is rather sensitive to the underlying input tuples, which may be imprecise and erroneous in nature. This paper investigates the sensitivity of answer ordering

returned from probabilistic database queries, aiming to find l basic tuples to which a certain answer ordering change is most sensitive.

Two typical kinds of probabilistic database queries are considered in the study [11,6]:

1) *probabilistic top-K queries*, aiming to find K answer tuples with the highest probabilities, since users are often interested in the answers with the highest probabilities;
2) *probabilistic top-K aggregation queries*, aiming to find K groups with the highest aggregate values, whose computation involves basic tuples' probabilities.

1.2 Related Work

Kanagal *et al.* [7] built a robust probabilistic query processing framework for sensitivity and explanation analysis of query results, they focused more on sensitivity analysis of each answer tuple. Despite the discussion was also extended to top-K queries, [7] considered a top-K query result as a set rather than a ranked list. In their work, for answer tuples t_1, t_2, t_3, t_4, t_5 in a top-5 query, the change from result1: $\langle t_1, t_2, t_3, t_4, t_5 \rangle$ to result2: $\langle t_5, t_4, t_3, t_2, t_1 \rangle$ is ignored, because it does not make changes to the top-5 set. However, the ordering of the elements in the answer list matters highly for certain applications, such as athletes ranking in Olympic Games, movie ranking at Oscar time, and mobile brand ranking in market share contest. In our work, we also consider the ranking inside the answer set, and regard result1 and result2 as completely different answers. The work by Soliman *et al.* [13] investigated the influence of uncertain scoring functions (not basic probabilistic tuples) on top-K join query results.

1.3 Contributions and Paper Organization

Different from the previous sensitivity analysis work in the database domain [7,13], this paper examines the ordering sensitivity of a top-K answer returned from probabilistic top-K (aggregation) queries to basic database tuples, like *which basic tuple's probability change can substantially alter the ranked answer list? which basic tuple's probability change will make a certain element top-ranked? which basic tuple's probability change will swap the positions of the firstly and secondly ranked elements? which basic tuple's probability change will make an answer tuple enter/drop out of the top-3 list?* etc.

The primary contributions of the paper are summarized as follows.

1. We categorize a variety of ordering sensitivity questions into *list*-oriented or *element*-oriented, and formulate the sensitivity analysis problem for answer ordering returned from probabilistic top-K (aggregation) queries.
2. We identify four basic modules in computing ordering-sensitivity, and develop a modular approach for ordering sensitivity computation in probabilistic databases.
3. We further give optimization strategies to improve the computational efficiency of sensitivity analysis.

Our experimental results on both synthetic and real data demonstrate the effectiveness and efficiency of the proposed solutions.

The remainder of this paper is organized as follows. We formulate the ordering-sensitivity analysis problem for probabilistic top-K (aggregation) query answers in Section 2, and provide a modular approach for efficient sensitivity computation in Section 3, followed by a performance study in Section 4. We conclude the paper with a brief discussion of future work in Section 5.

2 Problem Statement

Before the problem statement for sensitivity analysis of answer ordering in probabilistic databases, let's first review the probabilistic data model and corresponding probabilistic database queries.

2.1 Probabilistic Database Queries

Probabilistic Database Model. We adopt the tuple uncertainty probabilistic database model proposed by Dalvi *et al.* [4]. Each tuple t_i in a probabilistic database represents a probabilistic basic event e_i with probability $p(e_i)$. Assume all the probabilistic events are independent. Each result tuple r_i in a derived relation would be assigned a complex event expression ee_i called lineage which is a Boolean combination of the basic events corresponding to the basic tuples from which it was derived. Each of the basic operation in relational algebra (selection, projection, union, difference, and Cartesian product) has a corresponding event expression generating rule. [4] detailed the process of computing the result tuple event expressions and the equivalent possible worlds semantics. From the event expression ee_i of result tuple r_i, the probability $p(ee_i)$ can be computed based on the underlying basic events' probabilities. In the following, we shall use $p(t_i)$ and $p(e_i)$, $p(r_i)$ and $p(ee_i)$ interchangeably.

In this study, two typical kinds of probabilistic database queries are considered.

Probabilistic Top-K Queries. From a set of basic tuples $B = \{t_1, t_2, \cdots, t_n\}$ in the probabilistic database, a set of result tuples $S = \{r_1, r_2, \cdots, r_m\}$ is derived with complex event expressions $\{ee_1, ee_2, \cdots, ee_m\}$, whose probabilities $\{p(ee_1), p(ee_2), \cdots, p(ee_m)\}$ are computed according to the basic input tuples' probabilities. The probabilistic top-K query is to find K result tuples from S whose complex event expressions have the highest probabilities [11].

Probabilistic Top-K Aggregation Queries. In a probabilistic database, from a set of basic tuples $B = \{t_1, t_2, \cdots, t_n\}$, a set of result tuples $S = \{r_1, r_2, \cdots, r_m\}$ is derived, and further divided into w groups $GroupSet = \{G_1, G_2, \ldots, G_w\}$ according to the grouping attribute set A.

Each group $G \in GroupSet$ consists of a set of tuples $G = \{r_1, r_2, \cdots, r_{|G|}\}$, derived with the complex event expressions $\{ee_1, ee_2, \cdots, ee_{|G|}\}$ of probabilities $\{p(ee_1), p(ee_2), \cdots, p(ee_{|G|})\}$. Each tuple $r_i \in G$ has a score value v_i.

The expected aggregate value (sum, count, max, min, or avg) of group G, denoted as $F_{sum}(G)$, $F_{count}(G)$, $F_{max}(G)$, $F_{min}(G)$, $F_{avg}(G)$ respectively, can be further computed based on $p(ee_1), p(ee_2), \cdots, p(ee_{|G|})$ and score values $\{v_1, v_2, \cdots, v_{|G|}\}$ [6,7]. For the sum operator, $F_{sum}(G) = \sum_{i=1}^{|G|} p(ee_i) * v_i$. The count operator is just a special case of sum, where all the score values take the value 1, that is, $F_{count}(G) = \sum_{i=1}^{|G|} p(ee_i)$. Assume all the score values $\{v_1, v_2, \cdots, v_{|G|}\}$ are sorted in a non-increasing order. Then, for the max operator, $F_{max}(G) = \sum_{i=1}^{|G|} p(ee_i \wedge (\bigwedge_{1 \le j < i} \sim ee_j)) * v_i$. For the min operator, $F_{min}(G)$ can be computed similarly, except that the score values are sorted in a non-decreasing order. For the AVG operator, $F_{avg}(G) = \sum_{x_1, \cdots, x_{|G|}} \frac{\sum_{i=1}^{|G|} \chi_{x_i}(v_i)}{\sum_{i=1}^{|G|} x_i} * p(\bigwedge_{i=1}^{|G|} \chi_{x_i}(ee_i))$, where x_i takes value 0 or 1, and

$$\chi_{x_i}(v_i) = \begin{cases} 0 & \text{if } x_i = 0 \\ v_i & \text{if } x_i = 1 \end{cases}, \qquad \chi_{x_i}(ee_i) = \begin{cases} \sim ee_i & \text{if } x_i = 0 \\ ee_i & \text{if } x_i = 1 \end{cases}$$

All the groups in *GroupSet* are sorted by their expected aggregate values, with the top-K groups returned as the probabilistic top-K aggregation query answer.

2.2 Answer Ordering Change

For both probabilistic top-K queries and probabilistic top-K aggregation queries, the answer is an ordered list of K elements ranked based on certain ranking criteria. Without loss of generality, let $R = \langle r_1, r_2, \cdots, r_K \rangle$ be a top-K answer list with $Rank(r_1) \ge Rank(r_2) \ge \cdots \ge Rank(r_K)$. We use $R.r_i$ $(1 \le i \le K)$ to represent an answer element r_i, which can be either specified by its relative position in R like $r_i = R[i]$ or by its attribute value like $r_i.att = val$.

The ranking order of answer R is sensitive to perturbations of basic tuples' probabilities, e.g., some element may drop out of the top-K list, some new element may enter the top-K list, some element may swap the position with another one, etc. after the change of some basic tuples' probabilities. Table 1 illustrates two types of ordering changes (i.e., *list*-oriented and *element*-oriented) from a top-K answer list R to a new top-K answer list R', generally expressed via an ordering change predicate $P_{red}(R, R')$. We call the ordering change $P_{red}(R, R')$ happens if and only if $P_{red}(R, R') = TRUE$.

2.3 Measurement of Answer Ordering Change

To measure the ordering change from a top-K answer list R_1 to R_2, we extend the classic Spearman's footrule metric [5] to evaluate more highly the position changes at the top places than the bottom places.

In the following, for a list σ, we use S_σ to denote the set which contains all the elements in σ and call it the element set of σ, and $\sigma(x)$ to denote the position of element x in σ.

Classic Spearman's Footrule Metric. For two lists σ_1 and σ_2 with the same element set, the Spearman's footrule distance between σ_1 and σ_2 is defined as

$$D(\sigma_1, \sigma_2) = \sum_{x \in S_{\sigma_1}} |\sigma_1(x) - \sigma_2(x)|,$$

Table 1. Ordering change from R to R', where $R = \langle r_1, r_2, \cdots, r_K \rangle$ is the top-K answer list returned from a probabilistic top-K (aggregation) query, and $R' = \langle r'_1, r'_2, \cdots, r'_K \rangle$ is another top-K answer list from the same query after changing basic tuples' probabilities.

Category	**Ordering Change Predicate** $P_{red}(R,R')$	**Holding Condition**
List-Oriented	$lChange(R, [R'])$	R changes to R' (if R' is given), otherwise, R changes;
Element-Oriented	$eUpPosi(R, R', R.r, [\Delta p], [K])$	$R.r$ goes up Δp positions or enters top-K in R', i.e.,when Δp appears: $\exists i\ (1 \leq i \leq K)\ (R.r = R[i]) \wedge (R.r = R'[i + \Delta p])$, when K appears: $\forall i\ (1 \leq i \leq K)\ (\exists j\ (1 \leq j \leq K)\ (R.r \neq R[i]) \wedge (R.r = R'[j]))$;
	$eDownPosi(R, R', R.r, [\Delta p], [K])$	$R.r$ drops down Δp positions or drops out of top-K, i.e., when Δp appears: $\exists i\ (1 \leq i \leq K)\ (R.r = R[i]) \wedge (R.r = R'[i - \Delta p])$, when K appears: $\forall i\ (1 \leq i \leq K)\ (\exists j\ (1 \leq j \leq K)\ (R.r \neq R'[i]) \wedge (R.r = R[j]))$;
	$eRemainPosi(R, R', R.r)$	$R.r$'s position remains unchanged, i.e., $\exists i\ (1 \leq i \leq K)\ (R.r = R[i]) \wedge (R.r = R'[i])$;
	$eSwapPosi(R, R', R.r, R.r')$	$R.r$ and $R.r'$ swap the positions, i.e., $\exists i, j\ (1 \leq i, j \leq K)\ (R.r = R[i]) \wedge (R.r' = R[j]) \wedge (R.r = R'[j]) \wedge (R.r' = R'[i])$;
	$eKeepOrder(R, R', R.r, R.r')$	The order of $R.r$ and $R.r'$ remain the same, i.e., $\exists i, j\ (1 \leq i, j \leq K)\ (R.r = R[i]) \wedge (R.r' = R[j])$ $\exists i', j'\ (1 \leq i', j' \leq K)\ (R.r = R[i']) \wedge (R.r' = R'[j'])$ $(i > i' \rightarrow j > j') \vee (i < i' \rightarrow j < j')$;
	$eReverseOrder(R, R', R.r, R.r')$	The order of $R.r$ and $R.r'$ changes reversely, i.e., $\exists i, j\ (1 \leq i, j \leq K)\ (R.r = R[i]) \wedge (R.r' = R[j])$ $\exists i', j'\ (1 \leq i', j' \leq K)\ (R.r' = R'[i']) \wedge (R.r' = R'[j'])$ $(i > i' \rightarrow j < j') \vee (i < i' \rightarrow j > j')$.

which represents the summation of all the elements' position changes. For example, if $\sigma_1 = \langle b, c, a, d \rangle$, $\sigma_2 = \langle a, c, d, b \rangle$, the distance between σ_1 and σ_2 is $D(\sigma_1, \sigma_2) = |\sigma_1(a) - \sigma_2(a)| + |\sigma_1(b) - \sigma_2(b)| + |\sigma_1(c) - \sigma_2(c)| + |\sigma_1(d) - \sigma_2(d)| = 6$.

Extension of Spearman's Footrule Metric. We bind a weight to every two consecutive position change. For two lists σ_1 and σ_2 of the same n elements, $(n-1)$ weights $w_1, w_2, \ldots, w_{n-1}$ are introduced with $w_1 > w_2 > \ldots > w_{n-1} > 0$. Weight w_i measures the contribution of moving an element from position i to position $i+1$ or from position $i+1$ to position i to the distance of two lists. For any two positions i and j, let

$$C(i,j) = \begin{cases} \sum_{k=i}^{j-1} w_k & \text{if } i < j; \\ 0 & \text{if } i = j; \\ \sum_{k=j}^{i-1} w_k & \text{if } i > j \end{cases} \tag{1}$$

be the contribution of moving an element from position i to position j. For σ_1 and σ_2, the contribution of moving element x is $C(\sigma_1(x), \sigma_2(x))$. For the weighted Spearman's footrule metric, the distance between lists σ_1 and σ_2 of the same elements is defined as

$$D^w(\sigma_1, \sigma_2) = \sum_{x \in S_{\sigma_1}} C(\sigma_1(x), \sigma_2(x)). \quad (2)$$

For example, let $\sigma_1 = \langle b, c, a, d\rangle$ and $\sigma_2 = \langle a, c, d, b\rangle$. The introduced weights are w_1, w_2, w_3. Then, for the distance between σ_1 and σ_2, the contribution of moving element a is $C(\sigma_1(a), \sigma_2(a)) = w_1 + w_2$. Similarly, $C(\sigma_1(b), \sigma_2(b)) = w_1 + w_2 + w_3$, $C(\sigma_1(c), \sigma_2(c)) = 0$, $C(\sigma_1(d), \sigma_2(d)) = w_3$. The weighted distance between σ_1 and σ_2 is $D^w(\sigma_1, \sigma_2) = 2w_1 + 2w_2 + 2w_3$.

We now discuss modifications of the weighted Spearman's footrule metric D^w for the case when we only have top-K members of the ordering. Let R_1 and R_2 be two top-K lists, which may contain different sets of elements, i.e. $S_{R_1} \neq S_{R_2}$. The minimizing weighted Spearman's footrule distance $D^w_{min}(R_1, R_2)$ between R_1 and R_2 is defined to be the minimum value of $D^w(\sigma_1, \sigma_2)$, where σ_1 and σ_2 are any two lists with element set $S_{R_1} \cup S_{R_2}$ and $\sigma_1 \succeq R_1$, $\sigma_2 \succeq R_2$. Here, for two lists R and σ with $S_R \subset S_\sigma$, we use $\sigma \succeq R$ to denote $R(x) = \sigma(x)$ for all $x \in S_R$ and call σ an extension of R. Similarly, the maximizing weighted Spearman's footrule distance $D^w_{max}(R_1, R_2)$ and the averaging weighted Spearman's footrule distance $D^w_{avg}(R_1, R_2)$ can be defined by using the maximum and average value of $D^w(\sigma_1, \sigma_2)$ respectively. We have the following conclusion, with the proof omitted due to space constraint.

Proposition 1. *For any two top-K lists R_1 and R_2,*

$$D^w_{min}(R_1, R_2) = D^w_{avg}(R_1, R_2) = D^w_{max}(R_1, R_2) = D^w(\sigma_1, \sigma_2) \quad (3)$$

We also use $D^w(R_1, R_2)$ to denote $D^w_{min}(R_1, R_2)$, $D^w_{avg}(R_1, R_2)$ or $D^w_{max}(R_1, R_2)$ since they all have the same value. For example, if $R_1 = \langle a, b, c\rangle$, $R_2 = \langle b, a, d\rangle$, then $D^w(R_1, R_2) = D^w(\langle a, b, c, d\rangle, \langle b, a, d, c\rangle) = 2w_1 + 2w_3$ when the corresponding weights are w_1, w_2, and w_3 respectively.

2.4 Sensitivity of Answer Ordering Change

Definition 1. *Let $t \in B$ be a basic tuple in the probabilistic database. Let R be a top-K answer list returned from a probabilistic top-K (aggregation) query. Let R' be another top-K answer list from the same query after the change of t's probability. If the answer ordering change $P_{red}(R, R') = TRUE$ (listed in Table 1), we call the answer ordering change $P_{red}(R, R')$ is **sensitive** to basic tuple t.* □

Based on the distance measurement between two top-K answer lists, we can define the sensitivity degree of an answer ordering change to underlying basic tuples. Intuitively, when an ordering change happens, the less the deriving tuple's probability changes, the more sensitive it is.

Definition 2. *Let $t \in B$ be a basic tuple in the probabilistic database. Let R and R' ($R \neq R'$) be two top-K answer lists returned from a probabilistic top-K (aggregation) query before and after the change of t's probability. Assume the*

answer ordering change $P_{red}(R, R')$ is sensitive to basic tuple t. The ***sensitivity degree of basic tuple*** *t with respect to $P_{red}(R, R')$ is computed as follows.*

Case 1: when the change result R' is given,

$$Sensitivity(t) \mid_{P_{red}(R,R')} = \frac{1}{min|\Delta p(t)|} \mid_{P_{red}(R,R')=TRUE} \quad (4)$$

where $p(t)$ corresponds to the probability of tuple t, and $min|\Delta p(t)|$ returns the minimum change of t's probability that makes $P_{red}(R, R') = TRUE$ with $0 < |\Delta p(t)| < 1$.

Case 2: when the change result R' is omitted,

$$Sensitivity(t) \mid_{P_{red}(R,-)} = max_{R_x} \frac{D^w(R, R_x)}{min|\Delta p(t)|} \mid_{P_{red}(R,R_x)=TRUE} \quad (5)$$

where R_x is any possible ordering change derived from the original top-K answer list R, and $min|\Delta p(t)|$ returns the minimum change of t's probability that makes $P_{red}(R, R_x) = TRUE$ with $0 < |\Delta p(t)| < 1$. □

The sensitivity analysis of answer ordering in probabilistic databases can thus be formally stated as follows.

> Given a probabilistic top-K (aggregation) query answer R, the sensitivity computation of R's ordering change $P_{red}(R, R')$ is to find l basic tuples from the probabilistic database with the highest sensitivity degrees.

A declarative way for users to specify their ordering sensitivity analysis requirements is as follows:

`OS-ANALYZE [`l`]` ⟨`Ordering-Change-Predicate`⟩
`ANSWER-OF` ⟨`Query-Expression`⟩

where ⟨Ordering-Change-Predicate⟩ is given in Table 1, and ⟨Query-Expression⟩ is a probabilistic top-K (aggregation) query statement, whose answer ordering sensitivity is to be analyzed.

3 Sensitivity Computation

In this section, we present four computational modules for ordering change sensitivity analysis, and further develop optimization strategies to improve computational efficiency.

3.1 Computational Modules

The sensitivity computation of an answer ordering change $P_{red}(R, R')$ involves the following computational modules. First, for each answer element r in the answer list R, we need to examine how a basic tuple t's probability change affects its rank value. Second, since an answer ordering change corresponds to a series of answer tuples' swaps, for every two elements, we need to examine how a basic tuple t's probability change will swap their order. Third, the distance between

two orderings needs to be measured, based on which the sensitivity degrees of basic tuples can be computed. Fourth, we need to compute the sensitivity degree of a tuple.

Module 1 - Computing the effect of a tuple's probability change to an answer element We start with the probabilistic top-K query. The lineage of each result tuple r is an event expression ee which is a Boolean Formula of its deriving basic events. For each deriving basic event e (corresponding to an input tuple t), $p(ee)$ is a linear function of $p(e)$,

$$p(ee) = \alpha * p(e) + \beta \tag{6}$$

where α and β are functions of probabilities of other deriving basic events except e, and can be considered as constants with respect to $p(e)$. From Formula (6), we can derive $\frac{\partial p(ee)}{\partial p(e)} = \alpha$, which represents the change rate of $p(ee)$ with respect to $p(e)$. The linear property of $p(ee)$ with respect to $p(e)$ and the computing process of $\frac{\partial p(ee)}{\partial p(e)}$ can be found in [7].

Suppose that the probability value of basic event e corresponding to the input tuple t stored in the database is $p_0(e)$, and the corresponding probability value of result tuple r is $p_0(ee)$. Then, the relationship of $p(ee)$ and $p(e)$ can be considered as a straight line in a plane with equation

$$p(ee) = \frac{\partial p(ee)}{\partial p(e)}(p(e) - p_0(e)) + p_0(ee) \tag{7}$$

For input tuple t, there are K straight lines corresponding to the K result tuples. If e does not occur in the lineage of r, then $\frac{\partial p(ee)}{\partial p(e)} = 0$, and the corresponding straight line is $p(ee) = p_0(ee)$. When the probability of e changes in $[0, 1]$, each change of the result ranking corresponds to an intersection point of two of the K straight lines.

For the probabilistic top-K aggregation query, the average aggregation for each group(including sum, count, max, min, avg) is also a linear function of the deriving basic event probability. Hence, the idea that the intersection point of two straight lines corresponds to a change of the result ranked list also applies to the probabilistic top-K aggregation query.

Basic Approach. For the probabilistic top-K query, we adopt the approach proposed in [7] to compute $\frac{\partial p(ee)}{\partial p(e)}$.

For the probabilistic top-K aggregation query, the partial derivative of the average aggregation for each group G with respect to basic event probability $p(e)$ can be computed as follows.

SUM: $\frac{\partial F_{sum}(G)}{\partial p(e)} = \sum_{i=1}^{|G|} \frac{\partial p(ee_i)}{\partial p(e)} * v_i$.

COUNT: $\frac{\partial F_{count}(G)}{\partial p(e)} = \sum_{i=1}^{|G|} \frac{\partial p(ee_i)}{\partial p(e)}$.

MAX: $\frac{\partial F_{max}(G)}{\partial p(e)} = \sum_{i=1}^{|G|} \frac{\partial p(ee_i \wedge (\bigwedge_{1 \le j < i} \sim ee_j))}{\partial p(e)} * v_i$.

MIN: $\frac{\partial F_{min}(G)}{\partial p(e)} = \sum_{i=1}^{|G|} \frac{\partial p(ee_i \wedge (\bigwedge_{1 \le j < i} \sim ee_j))}{\partial p(e)} * v_i$.

AVG: $\frac{\partial F_{avg}(G)}{\partial p(e)} = \sum_{x_1, \cdots, x_{|G|}} \frac{\sum_{i=1}^{|G|} \chi_{x_i}(v_i)}{\sum_{i=1}^{|G|} x_i} * \frac{\partial P(\bigwedge_{i=1}^{|G|} \chi_{x_i}(ee_i))}{\partial p(e)}$.

Optimization. For a basic event e which occurs in result tuple r but not in the top-K results, if when $p(e)$ changes to 1, $p(r)$ cannot enter the top-K, or equivalently, $p_0(r) + \frac{\partial p(r)}{\partial p(e)}(1 - p_0(e)) < p_0(r_K)$, where r_K represents the K^{th} result element, then the probability change of e will not lead to a change of the result ranked list. This is equivalent to $\frac{\partial p(r)}{\partial p(e)} < \frac{p_0(r_K)-p_0(r)}{1-p_0(e)}$. Therefore, in the process of computing $\frac{\partial p(r)}{\partial p(e)}$, if $\frac{\partial p(r)}{\partial p(e)} < \frac{p_0(r_K)-p_0(r)}{1-p_0(e)}$, then the computing process for $\frac{\partial p(r)}{\partial p(e)}$ can be stopped, avoiding the computation of the exact value.

Module 2 - Computing the effect of a tuple's probability change to two elements' ordering

Basic Approach. For a probabilistic top-K query, for basic event e, the intersection point of the two straight lines corresponding to r_u and r_v with the following two equations $p(ee_u) = \frac{\partial p(ee_u)}{\partial p(e)}(p(e) - p_0(e)) + p_0(ee_u)$ and $p(ee_v) = \frac{\partial p(ee_v)}{\partial p(e)}(p(e) - p_0(e)) + p_0(ee_v)$ are represented as $\langle p(e) = p_0(e) + \frac{p_0(ee_u)-p_0(ee_v)}{\frac{\partial p(ee_v)}{\partial p(e)} - \frac{\partial p(ee_u)}{\partial p(e)}}, r_u, r_v \rangle$, meaning that r_u and r_v are swapped in the ranked list at $p(e) = p_0(e) + \frac{p_0(ee_u)-p_0(ee_v)}{\frac{\partial p(ee_v)}{\partial p(e)} - \frac{\partial p(ee_u)}{\partial p(e)}}$. For the case $p(e) \notin [0, 1]$ or $\frac{\partial p(ee_u)}{\partial p(e)} = \frac{\partial p(ee_v)}{\partial p(e)}$, the straight lines corresponding to r_u and r_v do not intersect and the positions of r_u and r_v do not change in the ranked list.

For the probabilistic top-K aggregation query, the process of computing intersection points is similar. For example, the sum aggregate corresponds to substituting ee_u and ee_v with $F_{sum}(G_u)$ and $F_{sum}(G_v)$ in the above computing process.

Optimization. For the basic event e, in the basic algorithm, to compute its influence, all the intersection points of all the straight lines corresponding to all the result tuples need to be computed. In fact, when computing the intersection points, only the straight lines corresponding to the top-K result tuples in which e does not occur and all the result tuples in which e occurs need to be considered, the intersection points produced by other straight lines will not correspond to a change of result ranked list, and can be ignored.

Module 3 - Measuring the distance between two top-K answer lists

Basic Approach. The distance between two ranked lists can be computed by using Formula (3) directly.

Optimization. Based on Module 2, which sorts all the intersection points according to their abscissas on the left-hand and right-hand side the $p_0(e)$, respectively, the distance computation can be further simplified. Suppose that when the probability of e changes from $p_0(e)$ to the right-hand side to $p'_1(e)$, $p'_2(e)$, $\cdots$, $p'_{s_1}(e)$ successively, the result ranked list changes to R'_1, R'_2, $\cdots$, R'_{s_1} accordingly; when the probability of e changes from $p_0(e)$ to the left-hand side to $p''_1(e)$, $p''_2(e)$, $\cdots$, $p''_{s_2}(e)$, the result ranked list changes to R''_1, R''_2, ..., R''_{s_2} accordingly.

When the probability of e changes from $p'_j(e)$ to $p'_{j+1}(e)$, the positions of result tuple x and y are inverted (in R'_j, x is ranked before y; in R'_{j+1}, x is

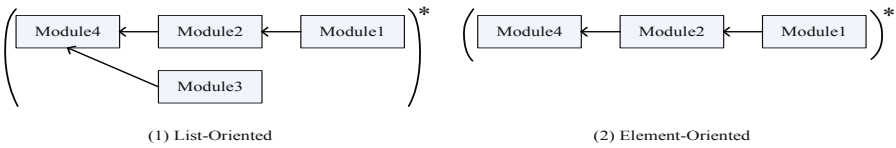


Fig. 3. The computation modules for ordering change sensitivity analysis

ranked after y). If we use x_0 and x_j to denote the position of x in R_0 and R'_j respectively, and y_0 and y_j the position of y in R_0 and R'_j respectively, then

$$D^w(R'_{j+1}, TopK_0) = D^w(R'_j, TopK_0) + \begin{cases} 2w_{x_j} & \text{if } x_0 \leq x_j \text{ and } y_0 \geq y_j; \\ -2w_{x_j} & \text{if } x_0 > x_j \text{ and } y_0 < y_j; \\ 0 & \text{if } (x_0 > x_j \text{ and } y_0 \geq y_j) \\ & \text{ or } (x_0 \leq x_j \text{ and } y_0 < y_j) \end{cases} \quad (8)$$

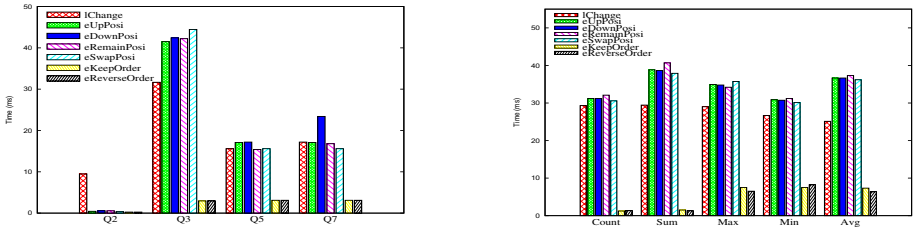
Hence, all the distances $D^w(R'_j, R_0)$ can be computed by combining Formula (8) and the initial condition $D^w(R'_0, R_0) = 0$. All the distances $D^w(R''_j, R_0)$ in the left hand side of $p_0(e)$ can be computed similarly. This optimization reduces the time complexity of computing distance between two top-K lists from $O(K)$ to $O(1)$.

Module 4 - Computing the sensitivity degree
Basic Approach. The sensitivity degree can be computed by using Formula (4) and (5).

Optimization. For the basic event e which only occurs in result r_i, the computation of sensitivity degree for e can be further simplified. Suppose that when the probability of e changes to 1, the probability of r_i changes to $p(r_i) = p_0(r_i) + \frac{\partial p(r_i)}{\partial p(e)}(1 - p_0(e))$ and r_i is changed to the i'th position; when the probability of e changes to 0, the probability of r changes to $p(r_i) = p_0(r_i) + \frac{\partial p(r_i)}{\partial p(e)}(-p_0(e))$ and r_i is changed to the i''th position. Then, $Sensitivity(e) = \frac{\partial p(r_i)}{\partial p(e)} \max(\max_{x=i'}^{i-1} \frac{1}{p_0(r_x) - p_0(r_i)} W(i,x),\ \max_{x=i+1}^{i''} \frac{1}{p_0(r_i) - p_0(r_x)} W(i,x))$, where $W(i,x) = 2\sum_{y=\min(i,x)}^{\max(i,x)-1} w_y$.

3.2 Sensitivity Computation

The sensitivity analysis for each ordering change first compute sensitivity degree for each basic tuple, and then select l tuples with the largest sensitivity degrees. We presents the modules for List-oriented and Element-Oriented ordering change sensitivity analysis and their invoke relationships in Figure 3. The arrow represents an invoke relationship between modules, and "*" represents a loop where each iteration computes the sensitivity degree of a basic tuple. The time costs of the basic algorithm for $eKeepOrder(R, R', R.r, R.r')$ and $eReverseOrder(R, R', R.r, R.r')$ are linear to the number of tuples in $R.r$ and $R.r'$. The time costs of the basic algorithm for all other five sensitivity



(*a*) Probabilistic Top-5 Queries

(*b*) Probabilistic Top-5 Aggregation Queries

Fig. 4. Answer-ordering analysis performance for different probabilistic top-K (aggregation) queries (DBSize=20M)

analyses are quadratic to the number of result tuples in the result ranked list. After the optimizations described in the above section, the number of tuples need to be considered are reduced substantially and the time costs are saved correspondingly, as shall be shown in the experimental results.

4 Evaluation

We evaluated our modular approach for sensitivity analysis in probabilistic databases on both synthetic and real dataset. All the relations were stored in MySQL. We implemented the experiments on a machine with a 2.70 GHz Core 2 Duo CPU and 2GB RAM.

4.1 Experiments on Synthetic Data

Datasets. We synthesized 20M to 100M TPC-H dataset augmented with tuple uncertainty for each tuple. The tuple existence probabilities were generated randomly in $[0, 1]$ in a uniform distribution.

Probabilistic top-K (aggregation) queries. We adopted TPC-H queries $Q2$, $Q3$, $Q5$, $Q7$, and $Q10$ as our probabilistic top-K queries, with all the aggregation constructs removed.

We modified TPC-H aggregation query $Q2$ as our base aggregation query, and implemented five probabilistic top-K aggregation queries, involving aggregate functions $Count$, Sum,Min, Max, and Avg.

Sensitivity analysis for the probabilistic top-K (aggregation) queries. For each query, we conducted seven ordering-sensitivity analyses on its query answer, belonging to different ordering change categories, as shown in Table 1.

Experimental results

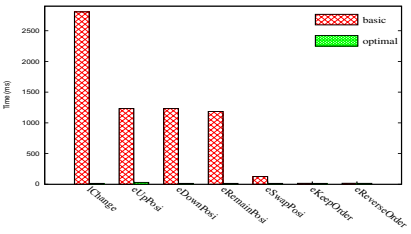
Answer-ordering sensitivity analysis is essential. Tables 2 and 3 show the maximum answer ordering changes(MAOC) resulting from probability change of a basic tuple for different top-5 and aggregation queries. We use the weighted Spearman's Footrule described in Section 2.3 to measure the answer ordering

Table 2. MAOC for Top-5 queries

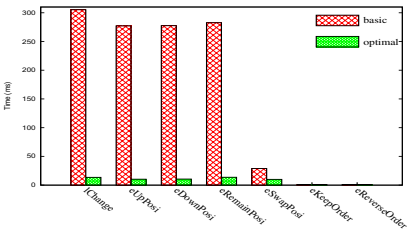
	Q2	Q3	Q5	Q7
MAOC	3.67	4.17	3.00	5.17

Table 3. MAOC Top-5 Aggregation Queries

	SUM	AVG	MAX	MIN
MAOC	3.00	3.67	4.58	3.00



(*a*) Top-5 Query $Q5$



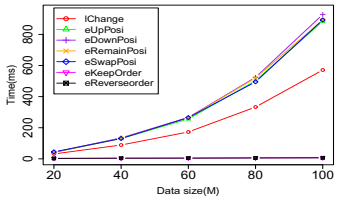
(*b*) Top-5 Aggregation Query (SUM)

Fig. 5. Ordering-Sensitivity Analysis Performance with/without Optimization (DB-Size=20M)

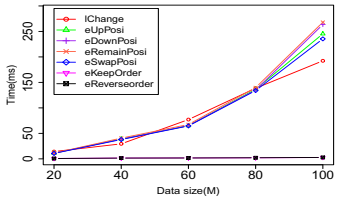
change, with the weights set to $\omega_i = \frac{1}{i}$ respectively. For example, for Q2, the maximum ordering change is: the first tuple changes to the second place, with a contribution 1.0; the second tuple changes to the third place, with a contribution 0.5; the third tuple changes to the fourth place, with a contribution 0.33; the fourth tuple changes to the first place, with a contribution 1.0 + 0.5 + 0.33 = 1.83; the overall weighted Spearman's Footrule distance is 3.67. We see that probability change of a basic tuple can lead to large answer ordering change for both top-K queries and top-K aggregation queries.

Answer-ordering sensitivity analysis performance for different probabilistic top-5 (aggregation) queries. Figure 4(a) and (b) show the analysis time for different top-5 and aggregation queries. The analysis time for top-K query $Q10$ in Figure 4(a) is around 60 times of $Q3$. It is long enough that we do not plot the analysis time of $Q10$ in Figure 4(a). From the figure, we can see that different times are needed for different kinds of sensitivity analyses for a query result. $eKeepOrder$ and $eReverseOrder$ take less time than the rest analyses. This is because the former only examines two involved elements, while the rest has to take care of more tuples in the answer list. Also, different queries need different analysis time due to different number of result tuples in total, despite only top-5 ones are returned.

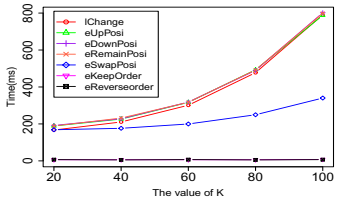
Answer-ordering sensitivity analysis performance with/without Optimization. Figure 5(a) and (b) show the analysis time of probabilistic top-5 query Q5 and probabilistic top-5 aggregation query SUM. We can see that the running time after taking our optimization strategies is much less than the basic algorithms without optimization for all the seven analyses. That is because considering all possible ordering changes in a large data set is quite time-consuming. A specified basic tuple may only influence a limited set of result tuples compared to the whole list. By optimization, intersections and basic events are ignored if they are found not to contribute to the change of top-K list by prejudging some conditions.



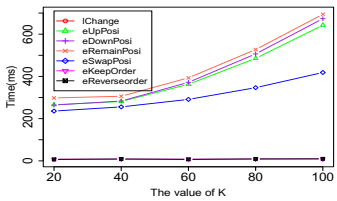
(*a*) Change of Database Size for Top-5 Query $Q5$



(*b*) Change of Database Size for Top-5 Aggregation Query SUM



(*c*) Change of K on Query $Q5$ (DBSize=100M)



(*d*) Change of K on Aggregation Query SUM (DBSize=100M)

Fig. 6. Scalability Performance of Ordering-Sensitivity Analysis

Besides, in computing the distance of two top-K lists, we adopt an incremental approach. In an ordered sequence of top-K lists $TopK_0, \cdots TopK_n$ stemming from the change of an basic event's probability, the distance $D(TopK_0, TopK_i)$ of $TopK_0$ and $TopK_i (i > 1)$ can be computed quickly by adding a small increment to the nearby distance result $D(TopK_0, TopK_{i-1})$. All these strategies can contribute to reduce the time costs of analysis.

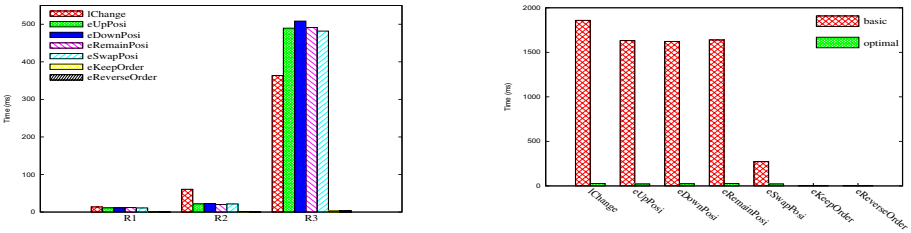
Scalability of Answer-ordering sensitivity analysis. We evaluated the scalability of our modular approach by varying the database size and value K. Figure 6(a) and (b) show the analysis time of probabilistic top-5 query $Q5$ and probabilistic top-5 aggregation query SUM which adopt all the optimizations under different database sizes. In both queries, $eKeepOrder$ and $eReverseOrder$ have a linear time complexity, and other analyses have a near linear time complexity. The basic tuples occurring in multiple result tuples' lineages lead to the derivation from time linearity. Figure 6(c) and (d) show the analysis time of probabilistic top-K query $Q5$ and probabilistic top-K aggregation query SUM with different K values. The experimental results show that the time cost is linear for $eKeepOrder$ and $eReverseOrder$ and near linear for all other queries as well.

4.2 Experiments on Real Data

Dataset. The real dataset integrates MovieLens data [10], with *actors* and *directors* information from IMDB movies database. MovieLens data records

Table 4. MAOC for Top-5 Queries on real data

	RQ1	RQ2	RQ3
MAOC	5.83	4.17	3.67



(*a*) Analysis on Real Data (*b*)Optimization on Real Data

Fig. 7. Experiments Results on Real Data

100,000 ratings (1-5) given by 943 users on 1682 movies. Based on the ratings from different users, we computed the probabilities that movies are liked by females and males and produced two probabilistic tables with 1682 rows in each table.

Queries and ordering-sensitivity analyses. Three queries are issued on the database:

$RQ1$: `Find the movies which are liked by females and males.`

$RQ2$: `Find the years where at least one movie is liked by females.`

$RQ3$: `Find the actors whose movies are liked by females and males.`

We also implemented the seven analyses on movie data.

Experimental results

Table 4 shows the maximum answer ordering changes(MAOC) resulting from probability change of a basic tuple for top-5 queries on real data. We use the weighted Spearman's Footrule described in Section 2.3 to measure the answer ordering change, with the weights set to $\omega_i = \frac{1}{i}$ respectively. We see that probability change of a basic tuple can lead to large answer ordering change for top-K queries on real data and the answer ordering sensitivity is essential.

Figure 7(a) shows the time costs of different sensitivity analyses for the three probabilistic top-5 queries($RQ1, RQ2, RQ3$). Figure 7 (b) compares the time costs of probabilistic top-5 query $RQ2$ before and after optimizations for different analyses. The effect of optimization in Figure 7(a) is more obvious than that in Figure 5(b). The reason is that many tuples in $RQ2$ appears only once, which can be processed quickly in our optimization strategy. For all the seven analyses, the time costs are significantly reduced.

In summary, the experimental results on both synthetic data and real data show that the optimization of our modular approach for answer ordering sensitivity analysis can improve the computational efficiency greatly.

5 Conclusion

In this paper, we studied the problem of sensitivity analysis for answer ordering in probabilistic databases. We categorized a variety of ordering sensitivity questions into *list*-oriented and *element*-oriented, and presented four basic processing modules and corresponding optimizations to resolve this problem. Evaluations on both synthetic data and real data verified that our solution can answer users' ordering change sensitivity analysis questions efficiently. In the future, we will extend our study to other probabilistic top-K queries (e.g. [12,8,9]) , uncertain K-nearest Neighbor queries(e.g. [2]), uncertain K-probable Nearest Neighbor queries (e.g. [3]) and other ordering concerned probabilistic queries proposed in the literature.

Acknowledgement. The work is supported by Tsinghua University Initiative Scientific Research Program, Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, National Natural Science Foundation of China (60773156, 61073004), Chinese Major State Basic Research Development 973 Program (2011CB302203-2), and Important National Science & Technology Specific Program (2011ZX01042-001-002-2).

References

1. Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: A system for data, uncertainty, and lineage. In: VLDB (2006)
2. Bernecker, T., Kriegel, H.-P., Mamoulis, N., Renz, M., Zuefle, A.: Scalable probabilistic similarity ranking in uncertain databases. Transactions on Knowledge and Data Engineering (2010)
3. Beskales, G., Soliman, M., Ilyas, I.: Efficient search for the top-k probable nearest neighbors in uncertain databases. In: VLDB (2008)
4. Dalvi, N., Suciu, D.: Efficient query evaluation on probabilistic databases. VLDB Journal (2007)
5. Fagin, R., Kumar, R., Sivakuma, D.: Comparing top k lists. SIAM Journal on Discrete Mathematics 17(1) (2003)
6. Jayram, T.S., Kale, S., Vee, E.: Efficient aggregation algorithms for probabilistic data. In: SODA (2007)
7. Kanagal, B., Li, J., Deshpande, A.: Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In: SIGMOD (2011)
8. Lian, X., Chen, L.: Probabilistic inverse ranking queries in uncertain databases. VLDB Journal (2011)
9. Lian, X., Chen, L.: Shooting top-k stars in uncertain databases. The VLDB Journal (2011)
10. Movielens data, `http://www.grouplens.data`
11. Ré, C., Dalvi, N., Suciu, D.: Efficient top-k query evaluation on probabilistic data. In: ICDE (2007)
12. Soliman, M.A., Ilyas, I.F., Chang, K.C.-C.: Top-k query processing in uncertain databases. In: ICDE (2007)
13. Soliman, M.A., Ilyas, I.F., Martinenghi, D.: Ranking with uncertain scoring functions: Semantics and sensitivity measures. In: SIGMOD (2011)

User Location Anonymization Method for Wide Distribution of Dummies

Ryo Kato[1], Mayu Iwata[1], Takahiro Hara[1],
Yuki Arase[2], Xing Xie[2], and Shojiro Nishio[1]

[1] Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka, Japan
[2] Microsoft Research Asia
Dan Ling Street, Haidian District, Beijing, China

Abstract. Advances in GPS (global positioning system) technologies have led to a variety of services utilizing the user's location becoming available. Since location information may reveal private information, preserving location privacy has become a significant issue. To preserve this privacy, we have proposed a user location anonymization method that generates dummies on the basis of the user's trajectory with pauses in movement. However, the previous work does not sufficiently satisfy users' requirements for location privacy because the locations of the generated dummies tended to gather in a small area as time passed. In this paper, we propose a user location anonymization method that always keeps dummies' locations in an area wide enough to anonymize the user's location. Our method periodically distributes dummies by moving them to an area where there were fewer users and dummies. We simulated the user's movement on a real map and verified that the proposed method was more effective than the previous one.

Keywords: Location-based services, Location privacy, GPS, Dummy.

1 Introduction

Location based services (LBSs) are becoming more common as a result of the growing popularity of mobile devices equipped with GPS (global positioning system) receivers. LBS providers provide a variety of services based on user locations, such as local searches, route planning, and location-based advertisements. However, location information may reveal private information, e.g., where an LBS user is living and to which school his or her children go. Beresford et al. [2] defined *location privacy* as the ability to prevent other parties from learning one's current or past location. They also warned that a system collecting users' locations potentially invaded their location privacy.

Numerous studies have been conducted to protect users' location privacy. User location anonymization methods using dummies are the representative approach[6][7][8] [10][11]. These methods generate dummies and send their locations together with the user's location to LBSs so that the LBS providers cannot distinguish between real and dummy locations. However, these existing methods did not consider physical constraints in a real environment; thus, their actual robustness in preserving privacy was

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 259–273, 2013.

questionable. The robustness of these methods depends strongly on how naturally the dummies behave. If the dummies behave unlike humans, such as moving at unreasonable speeds and being in the middle of seas or on mountain tops, it is easy to identify them as dummies. The traceability of locations is also a critical issue in a real environment. Since there are no perfect systems, it is always possible for user locations to be accidentally exposed. If this happens, the revealed location may further divulge locations where users have been and are going to be.

To solve these problems, we previously proposed user location anonymization methods using dummies (DumGrid[10] and Dum-P[6]) that take into consideration real-environment restrictions. Both methods generate dummies that move naturally like a user considering map information (locations where the user can exist) and user movement speed. Specifically, DumGrid generates dummies in a grid pattern around the user and moves them reactively according to the user's movement (i.e., the dummies tend to follow the user). However, DumGrid assumes a simplified mobility model in which the user keeps moving and does not stop. When we consider a more realistic mobility model in which the user often pauses to visit various attractions, it becomes more difficult to generate dummies that can move naturally. Dum-P generates dummies that move naturally while stopping at several locations like a real user on the basis of the user's movement with pauses in the movement. For Dum-P, to enable dummies to stop naturally, we assumed that the user movements (e.g., trajectory, pause time, and position) are known in advance. However, the dummies generated by Dum-P tend to gather in a small area as time passes because Dum-P first distributes dummies widely to anonymize the user's location and then makes them cross the user's path and those of other dummies to lower traceability as much as possible, (i.e., extensive distribution of dummies occurs just once while the crossing of dummies occurs many times). If dummies are located in a small area, an adversary could infer the user's approximate location even though it could not distinguish the user from the dummies. Considering this problem, Dum-P is inadequate for preserving location privacy regarding anonymity of the user's location.

In this paper, to supremely anonymize the user's location, we propose Dum-P-Cycle which is an extended method of Dum-P. Dum-P-Cycle consistently keeps dummies' locations in an area wide enough to anonymize the user's location on the basis of cycle-based dummy movements. Specifically, in the same manner as Dum-P, Dum-P-Cycle determines positions where and times when dummies should pause considering the known user movements and, based on them, generates the dummy movements that both move and stop. To distribute dummies widely all the time, Dum-P-Cycle sets the cycles in their movement plans and determines where and when the dummies should pause in the area with the smallest number of the user and dummies at the begging of each cycle, i.e., it periodically makes dummies move in order to spread out. Then, Dum-P-Cycle makes dummies cross the user's path and those of other dummies at positions where they pause in order to lower traceability as much as possible. As a result, dummies alternate between movement for wide distribution and movement for crossing on the basis of the cycles, i.e., both movements occurs many times. This movement pattern sufficiently satisfies the requirement for location privacy all the time. We also report

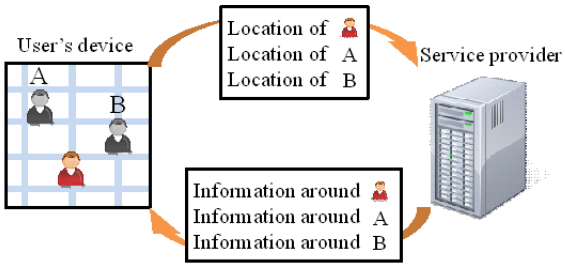


Fig. 1. Example of dummy-based approach

on an experiment conducted to evaluate this method in comparison with the previous methods, which shows the effect of the cycle.

The rest of the paper is organized as follows. Section 2 describes related work and Section 3 presents details of the proposed method. Section 4 describes our evaluation of the new approach and Section 5 concludes the paper with a summary and discussion of future work.

2 Related Work

There have been numerous studies on location privacy protection. We can categorize them into three approaches: 1) intermediating between the user and LBS, 2) transforming the user's location, and 3) generating dummy locations.

An example of an approach of the first type is the spatial cloaking[2][3][5]. This approach collects k users and sends the smallest region including those k user locations to an LBS server as a query instead of the specific user's location. This limits an adversary's probability of identifying the user's location to $1/k$. This approach needs to pool user's locations and assumes that there is a trusted third-party server between the users and the LBS server. However, it is difficult to deploy a completely safe third-party server in a real environment. Moreover, this approach fails to anonymize the user's location if there are not enough other users around.

An example of the second type is the obfuscation method[1][4][12]. This approach replaces a user's location with a near-by intersection or building to obscure the exact location. However, if there is no appropriate target around the user, the substitute location is far away from the user's location, which degrades the quality of a LBS services.

An example of the third type is the dummy-based approach[6][7][8][10][11]. As described in Section 1, this approach generates dummies around the user and sends their locations together with the actual user's location to an LBS server, as shown in Fig. 1. The LBS provider returns a list of information around the user and dummies. The user can get results related to his/her location by filtering out all the results corresponding to the dummy locations. Since only the user can distinguish his/her location from dummy locations, this approach can anonymize the user's location.

The dummy-based approach is reliable since it does not need an intermediate server/device for collection and only the user's device is involved in anonymizing the user's location. Moreover, this approach does not degrade the quality of service because the

results that the user gets are based on his/her precise location. Therefore, on the basis of the dummy-based approach, we previously proposed two dummy-based methods (DumGrid[10] and Dum-P[6]). DumGrid generates dummies around the user in a grid pattern considering physical constraints in a real environment. This method assumes a simplified mobility model in which users keep moving. Assuming a more realistic mobility model in which the user often pauses, Dum-P generates dummies on the basis of the known user's movements, where dummies can move naturally while stopping at several positions. However, Dum-P does not sufficiently consider the anonymity of the user's location, so its generated dummies gather in a small region as time passes. Therefore, Dum-P is inadequate to satisfy users' requirements for location privacy. Our new method (Dum-P-Cycle) generate dummies that sufficiently anonymize the user's location all the time by setting cycles of dummy movements and making dummies move to the area containing the fewest users and dummies in each cycle.

3 Proposed Method

This section first presents the requirements for location privacy protection and assumptions of the proposed method. Then it discusses a problem with our previous work and describes the proposed method (Dum-P-Cycle) in detail.

3.1 Requirement for Location Privacy

Anonymous Area. The anonymous area was defined following Lu et al. [8], as the size of the minimun convex covering all locations included in a service request. We define the anonymous area to measure the anonymity of the user's location as a criterion for evaluating how secure a user is in terms of location privacy. Fig. 2 (b) has an area with greater anonymity (higher-anonymity area) than Fig. 2 (a). If dummy locations are gathered in a small area like Fig. 2 (a), an adversary can infer the approximate location of the user. Therefore, we should satisfy the anonymous area size required by the user.

Our method locates dummies at positions where there are fewer of them (and the user) so that it can roughly form a grid and satisfy the required anonymous area size.

Traceability. We should also take into account the traceability of user locations. It might be possible to infer the trajectories of a user's movements from the location history that has been accumulated at an LBS provider on the basis of limitations on human movements. We define traceability as the ability to identify a user's trajectory by combining consecutive locations during a certain period. The traceability problem becomes serious especially when the user's locations are traceable; moreover, all previous (and possibly future) locations also become obvious. For example, when a user's locations are traceable, his/her trajectory is easily distinguishable from those of dummies, as shown in Fig. 3 (a). A simple but effective solution to lower traceability is to cross the trajectories of the user and dummies. This makes it difficult for an adversary to trace the user's trajectory, as shown in Fig. 3 (b).

Our method increases the chances the user and dummies cross by sharing positions where they pause.

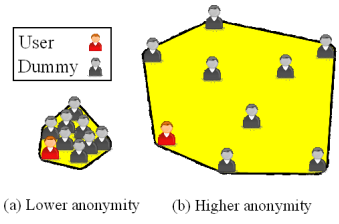


Fig. 2. Anonymous area

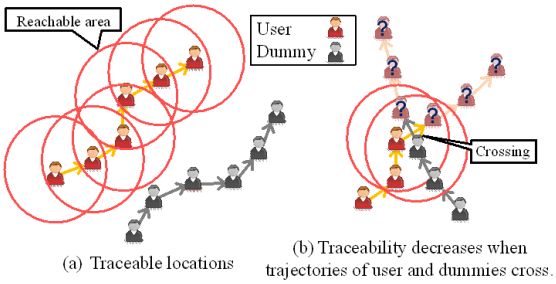


Fig. 3. Traceability

3.2 Assumption

We assume LBSs in which a user successively issues service requests (at constant or inconstant intervals) by sending user location information to the LBS provider and the LBS provider then returns information related to the user's current location.

We also assume a user mobility model in which the user is moving and stopping at several positions. In addition, we assume that the maximum speed, the minimum and maximum pause times, and the user's movement plans (e.g., trajectory, pause times, and positions) are known in advance. Although these assumptions might seem too strong (or unrealistic) in real situations, there are many situations in which we can know the user's movements in advance. For example, we can predict a user's future movements on the basis of certain additional information, e.g., his/her pre-registered plans of where and when he/she is going to go and the history of his/her trajectories. In this paper, we focus on how to generate dummies on the basis of user movements with a great deal of accuracy.

3.3 Our Previous Work (Dum-P)

Procedure. Dum-P first determines sets of positions where dummies should pause (*pause positions*) and times when they should start to pause at the pause positions (*pause start times*) on the basis of plans of a known user's movement and his/her requirements for the size of the anonymous area and the number of dummies. Then, Dum-P determines the dummies' movements such that they move toward the pause positions and arrive there at the pause start times.

Dum-P specifically carries out the following procedure by repeating steps (1) to (3) one by one for all the dummies. The first dummy is generated on the basis of the known user's movement, and subsequent ones are generated on the basis of both the generated dummies' and user's movements.

(1) Determine a set of the base pause position and the base pause start time of a new dummy to satisfy the required size of the anonymous area:
Dum-P determines one base pause position as a position in the area containing the fewest users and already-generated dummies exist and its base start time as the time when the smallest number first occurred in that area.

(2) Determine sets of shared pause positions and shared pause start times of the dummy to lower traceability:
Dum-P determines as many shared pause positions as possible such that the new dummy can share the pause positions of the user and already-generated dummies when they stop to cross dummies.
(3) Determine sets of mid-pause positions and mid-pause start times to ensure that the dummy's movement is natural:
Dum-P deterimines several mid-pause positions and mid-pause start times on the basis of the base and shared pause positions.

Briefly, the dummy moves while stopping at three types of pause positions: the base pause position, shared pause positions, and mid-pause positions.

Problem of Dum-P. We simulated user and dummy movements in Dum-P in order to make clear its problem. In this evaluation, we set the service cycle as 180[s], the number of dummies as 16, and the anonymous area requirement as 1600^2[m^2]. To measure the performance of Dum-P in terms of anonymity satisfaction and traceability reduction, we used the following evaluation metrics.

- **Anonymous area achieving Ratio-Count (AR-Count)**
  This metric aims to measure the satisfaction rate of the anonymous area requirement. Specifically, we counted the number of service requests when the size of the anonymous area that is the minimum convex covering all locations of dummies and the user satisfied the required anonymous area size in all service requests issued during the simulation time. AR-Count is defined as the ratio of the number of satisfied service requests to all the service requests.
- **Mean Time to Confusion (MTC)**
  Each queried location has a probability of being the user's location. When the user's location is accidentally identified, e.g., by a report of sighting in the real environment, the probability of the location being the user's location is 1. We defined the stochastic transition of the possibility as follows.

  When a location with probability α of being the user's location and another location with probability β are located in an area that is reachable from both their previous locations, these two locations cannot be distinguished. In this situation, the probability of both locations being the user's is $\frac{\alpha+\beta}{2}$. To measure the traceability in our evaluation, we use Mean Time to Confusion (MTC) defined in [9]. MTC is defined as the mean time necessary to anonymize the user's location from accidental revelation by the LBS provider. Every time a service request is issued, we calculate the entropy of the probability of being the user's location by $H = -\sum_{i \in \mathbf{D}} p_i \log p_i$, where p_i is the probability of location i being the user's location and D is the set of all locations corresponding to the user and all dummies. In our evaluation, we assumed that the user's location is sometimes revealed by the LBS provider. We defined MTC as the mean time period from the time when H becomes 0 (when the user's location is revealed) to the time when H exceeds 1 (when we can regard the user's location as being anonymized). A smaller MTC means lower user traceability.

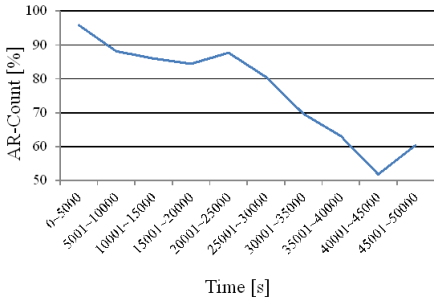


Fig. 4. Time course of AR-Count (Dum-P)

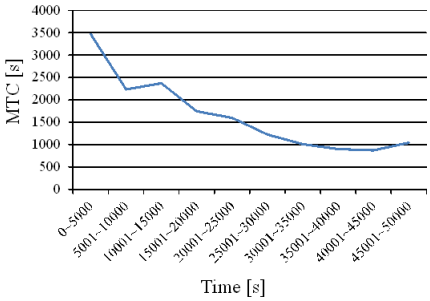


Fig. 5. Time course of MTC (Dum-P)

Time course results for AR-Count and MTC are shown in Figs. 4 and 5, respectively. We can see that, as time passed, both AR-Count and MTC degraded, i.e., dummies gathered in a small area, whereas they were located in a wide area at earlier times. This is because Dum-P determines one base pause position for each dummy as a position in the area containing the fewest users and dummies and its base pause time as the time when the smallest number first occurred in that area. Thus, most dummies tend to move towards the base pause position at a relatively early time and then move towards the shared pause positions to cross the user and other dummies during the remaining time, i.e., their areas of activity shrink as time passes. Therefore, Dum-P cannot fully satisfy the required anonymous area size.

If the anonymous area including the locations of the user and dummies is small, an adversary could easily infer the user's rough location. This is a significant issue in preserving the user's location privacy. For example, in this situation, the approximate location of the user's destination (e.g., user's house or office) could be roughly guessed. In particular, it becomes difficult (or impossible) to satisfy the user's requirement when the required anonymous area size is larger.

3.4 Our Approach

To solve the problem of Dum-P described in Section 3.3, we propose Dum-P-cycle, which is an extended version of Dum-P. Dum-P-Cycle generates dummies that pause at several positions by determining their pause positions on the basis of the known user's movement in the same way as Dum-P. However, whereas Dum-P determines one base pause position, Dum-P-Cycle determines multiple ones.

Design Policy. To prevent dummies from being located in a small area, Dum-P-Cycle sets cycles in their movement plans. For each cycle, Dum-P-Cycle determines a base pause position and base pause start time, i.e., a dummy's movement generated by Dum-P-Cycle contain multiple base pause positions. As a result, dummies move iteratively to the base pause position in order to expand the anonymous area at about the time when each cycle starts. Note that, it is important to set the same cycle length for all of the dummies. If we were to set different cycle lengths, some dummies would move to base pause positions to expand the anonymous area size (try to spread out), while others would move to shared pause positions to cross the user and other dummies (try to gather

together). This decreases the opportunity for dummies to cross them because the distance between them becomes large. By setting the same cycle length for all dummies, we ensure that all dummies move to the base pause position at about the same time and move to shared pause positions, i.e., they alternate between movement for wide distribution and movement for crossing. This movement pattern cannot only lower traceability but also expand the anonymous area effectively.

Procedure. Dum-P-Cycle first sets cycles in the movement of a new dummy and determines pause positions and pause start times on the basis of the known user's movement and the generated dummies' movements. Dum-P-Cycle specifically operates by repeating steps (1) to (5) one by one for the required number of dummies.

(1) Set the total travel time of a new dummy as that of the user.
(2) Divide the total travel time of the dummy into cycles (cycle lengh T).
(3) Determine sets of base pause positions and base pause start times of the dummy for each cycle.
(4) Determine sets of shared pause positions and shared pause start times of the dummy.
(5) Determine sets of mid-pause positions and mid-pause start times of the dummy.

Note that we added steps (1) and (2), extended step (3) corresponding to step (1) of Dum-P (described in Section 3.3), and use steps (4) and (5) in the same way as steps (2) and (3) of Dum-P. The number of base pause positions included in a dummy's movement is $\frac{\text{total travel time of the user}}{\text{cycle length } T}$. Dum-P-Cycle determines shared pause positions and mid-pause positions in the same way as Dum-P. Details of how to determine shared pause positions and mid-pause ones are described in [6].

As described in step (3) of the procedure, Dum-P-Cycle extends Dum-P's method of determining base pause positions and base pause start times. Algorithm 1 shows the algorithm for determining base pause positions and base pause start times of nth dummy. It specifically determines them by repeating steps (i) to (iv) one by one for the number of cycles.

(i) Determine a pause start time for the kth cycle of the new dummy.
(ii) Make a grid containing the user and already-generated dummies for the kth cycle.
(iii) Count the number of the user and generated dummies located in each grid cell.
(iv) Determine a base pause position.

In step (i), Dum-P-Cycle determines a base pause start time before or after the kth cycle starts, i.e., at around kT (T is cycle length). If we were to determine the base start time as just kT, all dummies would stop at the same time, which would cause unnatural movement of dummies. Therefore, we determine the base pause start time as a random time within a certain period around kT for each dummy. In step (ii), we make the grid containing the user's location and the locations of dummies that have already been generated at the time when the kth cycle started (kT), as shown in Fig. 6. The grid is a square having three $\times$ three grid cells. The width of the grid is $\sqrt{S}$ to satisfy the required size of the anonymous area, S. The center of the grid is set to the average position of the user's location and the already-generated dummies' locations. Then, in step (iii), on the basis of the grid made in step (ii), we count the number of the user and generated dummies in each grid cell at kT. The area with the fewest user

Algorithm 1 Determining sets of base pause positions and base pause start times of nth dummy

input: a list of a user and generated dummies' movements $D = \{D_0, \ldots, D_{n-1}\}$ (sets of their pause positions, pause start times and pause durations), possible pause positions based on the map information $P = \{P_0, \ldots, P_m\}$, cycle lengh T, simulation end time t_{end}
output: sets of pause positions and pause start times PP of nth dummy

$k \leftarrow 0$
repeat
  $t \leftarrow kT$
  $t_{base} \leftarrow t+$ random($-a$,$+a$) // determine kth base pause start time
  generate a grid with 3×3 cells $G = \{G_0, \ldots, G_8\}$ around the center of positions of D at t
  for $i = 0$ to 8 **do**
    $G_i.exist_t \leftarrow$ the number of D within G_i
  end for
  if $k = 0$ **then**
    //determine first base pause position
    $G_{base} \leftarrow G_i$ with min($G_0.exist_t$, ..., $G_8.exist_t$)
    $p_{base} \leftarrow$ random(P in G_{base})
  else
    //determine subsequent base pause position
    $p_{base} \leftarrow NULL$
    repeat
      if $p_{base} \neq NULL$ **then**
        append p_{base} to p_{false} // p_{false} is a list of unreachable pause positions
        if the number of P in G_{base} = the number of p_{false} **then**
          append G_{base} to G_{false} // G_{false} is a list of grid cell where all pause positons are unreachable
          $p_{false} \leftarrow 0$
        end if
      end if
      $G_{base} \leftarrow G_i$ with min($G_0.exist_t$, ..., $G_8.exist_t$; except G_{false})
      $p_{base} \leftarrow$ random(P in G_{base}; except p_{false})
    until (p_{base} is within the reachable area from a pause position with the latest pause start time in PP of nth dummy) OR (the number of cells in $G_{false} = 9$)
  end if
  if the number of cells in $G_{false} \neq 9$ **then**
    $PP \leftarrow <p_{base}, t_{base}>$
  end if
  $k \leftarrow k + 1$
until $kT > t_{end}$
return PP

and generated dummies can be identified at kT. If there are multiple areas containing the fewest users and generated dummies, we select one area randomly. Finally, in step (iv), we determine a base pause position as a position in the area determined in step (iii). When determining the second and subsequent base pause positions, it is important whether or not the dummy in the process can reach from the base position already determined in the $k-1$th cycle to the base pause position of the kth cycle to ensure that the dummy moves naturally like a user. If the dummy in the process cannot reach the kth base pause position at the user's moving speed from $k-1$th base pause position before kth base pause start time, the area for kth base pause position is determined as the area containing the second-fewest user and dummies. If it still fails, the next fewest area is examined repeatedly.

Briefly, the dummy moves while stopping at some base pause positions, some shared pause positions, and some mid-pause positions. An example of how a dummy's movements are determined by Dum-P-Cycle is outlined in Fig. 7. In this figure, we set the

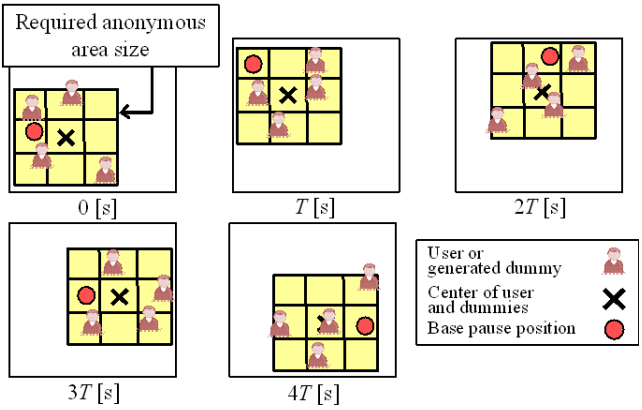


Fig. 6. Example of determining base pause positions and base pause start times on the basis of the grid (cycle length T)

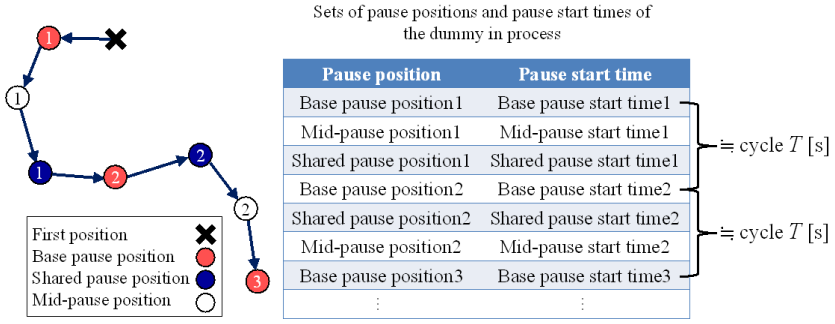


Fig. 7. Example of dummy's movements generated by Dum-P-Cycle (cycle T)

cycle length as T[s]. The time between one base pause start time and the next one is about the same as T[s].

4 Evaluation

In this experiment, we aimed to evaluate the robustness of our method from the quantitative perspective (i.e., statistical analysis). We compared our method with our previous methods[6][10].

4.1 Setting for Evaluation

We simulated human movements in Kyoto, Japan using a network simulator MobiREAL[1]. We set positions where a user and dummies can stop at every 50 [m] on roads. The parameters and their values used in our evaluation are summarized in Table 1. We used AR-Count and MTC for evaluation metrics (explained in Section 3.3).

[1] http://www.mobireal.net

Table 1. Parameters used in the experiment

Parameter	Value
Simulation time [s]	50000
Service cycle [s]	180
Moving speed [m/s]	1.30
Area size [m^2]	15200^2
Number of dummies	16, 25
Maximum pause time [s]	600
Minimum pause time [s]	60
Anonymous area requirement [m^2]	1000^2, 1100^2,..., 2000^2
Cycle length [s]	5000, 10000,..., 30000

4.2 Methods Used for Comparison

- **DumGrid**
  The method proposed in [10], which assumes that the user's movement cannot be predicted. This method arrays dummies around the user in a grid formation to react to the user's movement in order to achieve the required anonymous area size. It also makes dummies cross with the user's route to reduce the traceability of the user's location. It should be noted that, in this method, dummies cannot naturally pause like the user but do pause at unnatural positions where the user does not pause. Thus, it is easy to distinguish the user among dummies by visual observation. However, we ignored this problem in this evaluation.
- **Dum-P** [6]
- **Dum-P-Cycle (proposed method)**

4.3 Evaluation Results

AR-Count. AR-Counts with various anonymous area requirements, two different numbers of dummies (N = 16 and 25), and a particular cycle length to determine the base pause start times (T = 20000[s]) are shown in Fig. 8. The results indicate that as the anonymous area requirement increases, AR-Count becomes smaller for all of the methods. This shows a drawback of the traceability reduction process, which occasionally shrinks the anonymous area to cross the user and dummies.

The AR-Count for Dum-P-Cycle is larger than those for previous methods (DumGrid and Dum-P). The average difference between AR-Count of Dum-P-Cycle and that of DumGrid is 27.8% and that between AR-Count of Dum-P-Cycle and that of Dum-P is 31.8% when N = 16. In DumGrid, the positions of dummies are often changed from their ideal locations in a grid owing to geographical restrictions or crossing, so the anonymous area requirement often cannot be satisfied. In Dum-P, dummy positions also cannot satisfy the anonymous area size because dummies move to expand the anonymous area size only in early times, and then they gradually gather in a small area to lower traceability. On the other hand, Dum-P-Cycle determines dummies' movements that contain some base pause positions on the basis of the known user's movement. Therefore, the dummies generated by Dum-P-Cycle are not affected by geographical restrictions and they move to satisfy the anonymous area on several occasions through the simulation time. This result suggests that Dum-P-Cycle can achieve a larger anonymous area by setting some base pause positions on the basis of cycles.

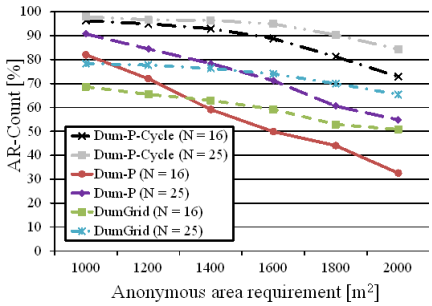


Fig. 8. AR-Count with various anonymous area requirements (T = 20000[s])

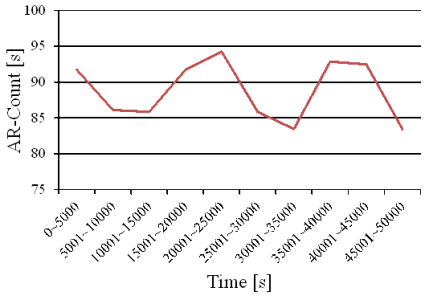


Fig. 9. Time course of AR-Count (Dum-P-Cycle, N = 16, T = 20000[s], anonymous area requirement 1600^2[m^2])

Comparing between N = 16 and N = 25, for all cases of anonymous area requirements, AR-Counts of DumGrid, Dum-P, and Dum-P-Cycle are larger for N = 25 than for N = 16. For Dum-P-Cycle, the average difference in AR-Count between them is 5.7%. This is because when the number of dummies is large, the anonymous area can increase easily.

Additionally, we investigated how the anonymous area of Dum-P-Cycle changes over time. The time course of AR-Count of Dum-P-Cycle, where N = 16, T = 20000[s], and the anonymous requirement is 1600^2 [m^2] is shown in Fig. 9. The result in Fig. 9 shows that in the simulation, AR-Count of Dum-P-Cycle fluctuated compared with that of Fig. 4. We can see large AR-Count around the cycles; 0[s]...5000[s], 20001[s]... 25000[s] and 40001[s]...45000[s]. This is because Dum-P-Cycle sets several base pause positions on the basis of the cycles throughout the simulation time. This result confirms that Dum-P-Cycle achieves high anonymous area size all of the time, so it solves the problem with Dum-P described in section 3.3.

MTC. MTCs for various anonymous area requirements where N = 16 and 25 and T = 20000[s] are shown in Fig. 10. The results show as the anonymous area requirement decreased, MTCs of all methods decreased. This is because when the anonymous area requirement is small, dummies are located closer to the user and the user's location is within their reachable areas in most cases, i.e., there is basically no need to intentionally cross the user and dummies.

MTCs of Dum-P-Cycle are slightly larger than those of Dum-P, i.e., Dum-P indicates slightly lower traceability than Dum-P-Cycle. The average difference in MTC between Dum-P-Cycle and DumGrid is 1768[s] and that between Dum-P-Cycle and Dum-P is 448[s] for N = 16. This is because Dum-P-Cycle sets more base pause positions than Dum-P, so the dummies generated by Dum-P-Cycle tend to be located in a larger area and have less opportunity to cross than those generated by Dum-P. On the other hand, for all cases of anonymous area requirements and numbers of dummies (N = 16 and 25), MTCs of Dum-P-Cycle were much lower than those of DumGrid. This result shows the effectiveness of aggressively reducing the traceability in Dum-P-Cycle. Dum-P-Cycle determines as many shared positions as possible to make dummies cross with the user and other dummies. On the other hand, in DumGrid, it is difficult

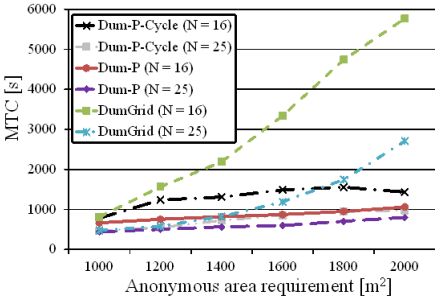


Fig. 10. MTCs with various anonymous area requirements (T = 20000[s])

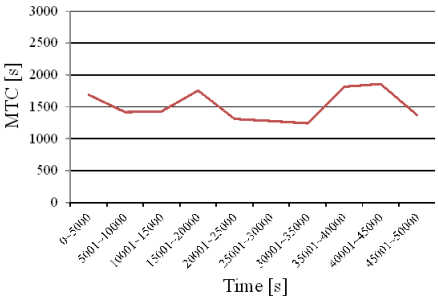


Fig. 11. Time course of MTC (Dum-P-Cycle, N = 16, T = 20000[s], anonymous area requirement 1600^2[m^2])

to make dummies cross naturally and often because DumGrid makes them cross only when some dummies move ahead of the user so that a grid formation that satisfies the anonymous area requirement can be kept. The movement directions of dummies also affect the difference between MTCs with Dum-P-Cycle and those with DumGrid. Specifically, dummies move in various directions in Dum-P-Cycle while they tend to move in parallel with the user's moving direction keeping a certain distance in DumGrid. Therefore, dummies generated by DumGrid have fewer chances to be located in the user's reachable area than those generated by Dum-P-Cycle.

Comparing between N = 16 and N = 25, for all cases of anonymous area requirements, the MTCs of all methods for N = 25 are smaller than those for N = 16. For Dum-P-Cycle, the average difference between them was 549[s]. This is because when the number of dummies is large, dummies are located closer to the user and the user's location is within their reachable areas in more cases, which increases the chances of crossing with dummies.

Additionally, we investigated how MTC of Dum-P-Cycle changes as time passes. The time course of MTC of Dum-P-Cycle, where N = 16, T = 20000[s] and the anonymous requirement is 1600^2 [m^2], is shown in Fig. 11. These results show that MTCs of Dum-P-Cycle fluctuated during the simulation compared with that in Fig. 5. We can see that MTCs increased around the cycles; 0[s]...5000[s], 20001[s]...25000[s] and 40001[s]...45000[s]. This is because Dum-P-Cycle determines shared pause positions aggressively for the whole simulation time except around the starts of cycles. This result confirms that Dum-P-Cycle lowers traceability all the time and solves the problem with Dum-P described in Section 3.3.

Effect of Cycle. Finally, we investigated the effect of cycle lengths in determining base pause start times on AR-Count and MTC. The results on AR-Count and MTC, where the anonymous requirement was 1600^2[m^2] are shown in Figs. 12 and 13, respectively. The results in Fig. 12 show that AR-Count became larger as the cycle length became longer (although the shorter the cycle length, the more often dummies move to base pause positions, i.e., they move to expand the anonymous area size). This is because Dum-P-Cycle determines base pause positions *within* a grid with the size of required anonymous area and determines base pause positions in the grid. When the cycle length

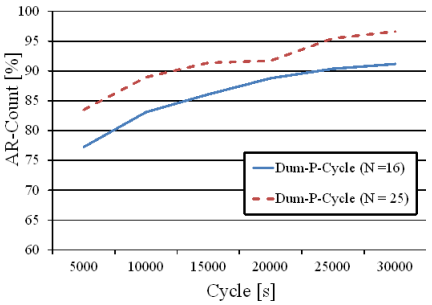


Fig. 12. AR-Count with various cycle lengths (anonymous area requirement $1600^2[\mathrm{m}^2]$)

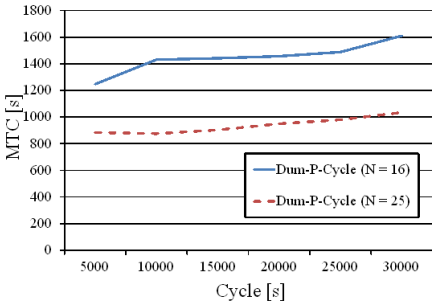


Fig. 13. MTC with various cycle lengths (anonymous area requirement $1600^2[\mathrm{m}^2]$)

is short, Dum-P-Cycle determines too many base pause positions like T = 5000[s]. In this situation, dummies tend to always be located in the area that satisfies the required anonymous area size. Thus, it is unlikely that the anonymous area can achieve the user's requirements.

The results in Fig. 13 show that MTCs became larger as the cycle length became longer (although with a shorter cycle length, dummies have more base pause positions, i.e., fewer shared pause positions). This is because the anonymous area formed by dummies in Dum-P-Cycle becomes small when the cycle length is short owing to the grid, as described above. When Dum-P-Cycle determines too many base pause positions like every 5000[s], the distance between the user and dummies and that between dummies are short. In this situation, the chances of crossing decrease because incidental crosses (i.e., not occurring as a result of setting shared pause positions) often happen.

As a whole, when the cycle length is long, Dum-P-Cycle can easily satisfy the required anonymous area, while it has difficulty in lowering traceability. On the other hand, when the cycle length is short, Dum-P-Cycle can easily lower traceability, while it has difficulty in satisfying the required anonymous area. Therefore, Dum-P-Cycle can meet the user's requirement regarding anonymous area or regarding traceability if we adjust the cycle length, i.e., if the user cares more about the anonymous area than about traceability, we should set a large cycle length.

5 Conclusions

We proposed a method (Dum-P-Cycle) to anonymize a user's location on the basis of the user's known movements with pauses when using LBSs with mobile devices. Dum-P-Cycle generates dummies that move naturally while stopping at several positions like the user aiming at a wide distribution of dummies to preserve the user's location privacy all the time. To anonymize the user's location, it makes dummies stop periodically at several positions having fewer user and dummies. To lower the traceability, it makes the user and dummies cross with each other at positions where they pause.

We simulated the user's movement on a real map and verified the effectiveness of Dum-P-Cycle in comparison with our previous methods (DumGrid[6] and Dum-P[10]). We found that Dum-P-Cycle could achieve a larger anonymous area and lower

traceability than DumGrid. Additionally, we confirmed that Dum-P-Cycle could always distribute dummies in a wide area and sufficiently anonymize the user's location compared with Dum-P.

As part of our future work, we plan to conduct a user experiment to evaluate the robustness of our method against humans (i.e., visual observation). Additionally, in order to make our approach more realistic, we plan to extend our method to make dummies react to unpredicted user's movement which does not completely follow the known one.

Acknowledgement. This work was partially supposed by Grant-in-Aid for Challenging Exploratory Research (24650038) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and the Microsoft Research Core Project.

References

1. Ardagna, C.A., Cremonini, M., Damiani, E., di Vimercati, S.D.C., Samarati, P.: Location privacy protection through obfuscation-based techniques. In: Proc. Int'l Conf. on Data and Applications Security, pp. 47–60 (July 2007)
2. Beresford, A.R., Stajano, F.: Location privacy in pervasive computing. In: Proc. Int'l Conf. on Pervasive Computing, pp. 46–55 (March 2003)
3. Bettini, C., Mascetti, S., Wang, X.S., Jajodia, S.: Anonymity in location-based services, towards a general framework. In: Proc. Int'l Conf. on Mobile Data Management, pp. 69–76 (May 2007)
4. Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: Proc. Int'l Conf. on Pervasive Computing, pp. 152–170 (May 2005)
5. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: Proc. Int'l Conf. on Mobile Systems, Applications, and Services, pp. 31–42 (May 2003)
6. Kato, R., Iwata, M., Hara, T., Suzuki, A., Arase, Y., Xie, X., Nishio, S.: A dummy-based anonymization method based on user trajectory with pauses. In: Proc. Int'l Conf. on Advance in Geographic Information Systems, pp. 249–258 (November 2012)
7. Kido, H., Yanagisawa, Y., Satoh, T.: An anonymous communication technique using dummies for location-based service. In: Proc. Int'l Conf. on Pervasive Service, pp. 88–97 (July 2005)
8. Lu, H., Jensen, C.S., Yiu, M.L.: PAD:Privacy-area aware, dummy-based location privacy in mobile services. In: Proc. Int'l Workshop on Data Engineering for Wireless and Mobile Access, pp. 16–23 (June 2007)
9. Shokri, R., Freudiger, J., Jadliwala, M., Hubaux, J.P.: A distortion-based metric for location privacy. In: Proc. ACM Workshop on Privacy in the Electronic Society, p. 6 (November 2009)
10. Suzuki, A., Iwata, M., Arase, Y., Hara, T., Xie, X., Nishio, S.: A user location anonymization method for location based services in a real environment. In: Proc. Int'l Conf. on Advance in Geographic Information Systems, pp. 398–401 (November 2010)
11. Yanagisawa, Y., Kido, H., Satoh, T.: Location traceability of users in location-based services. In: Proc. Int'l Conf'. on Mobile and Ubiquitous Systems Workshops (July 2006)
12. Yiu, M., Jensen, C., Huang, X., Liu, H.: Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In: Proc. Int'l Conf. on Data Engineering, pp. 366–357 (April 2008)

An XML-Based Policy Model for Access Control in Web Applications

Tania Basso[1], Nuno Antunes[2], Regina Moraes[1], and Marco Vieira[2]

[1] State University of Campinas (UNICAMP), Campinas, Brazil
{regina,taniabasso}@ft.unicamp.br
[2] University of Coimbra (UC), Coimbra, Portugal
{nmsa,mvieira}@dei.uc.pt

Abstract. Organizational Information Systems (IS) collect, store, and manage personal and business data. Due to regulation laws and to protect the privacy of users, clients, and business partners, these data must be kept private. This paper proposes a model and a mechanism that allows defining access control policies based on the user profile, the time period, the mode and the location from where data can be accessed. The proposed policy model is simple enough to be used by a business manager, yet it has the flexibility to define complex restrictions. At runtime, a protection layer monitors data accesses and enforces existing policies. A prototype tool was implemented to run an experimental evaluation, which showed that the tool is able to enforce access control with minimal performance impact, while assuring scalability both in terms of the number of users and the number of policies.

Keywords: access control, policy, data privacy, security.

1 Introduction

Nowadays, Internet has providing a wide range of services, such as e-commerce, banking, financial services, among others. Usually, to use these services, users, customers, and business partners need to provide private information such as addresses, social security IDs, and credit card numbers. The holders of these data have the obligation and responsibility to protect them, which raises special privacy concerns.

Privacy can be defined in many ways, but Bertino et al. [1] address it from the point of view of controlled information release, thus defining privacy as "*the right of an entity to be secure from unauthorized disclosure of sensible information that are contained in an electronic repository*". This is also the definition we adopt in the present work.

A well-known approach for protecting privacy in scenarios based on web applications and services is to manage data using access control policies. An access control policy can be defined as "*a set of rules by which human users, or their representatives, are authenticated and by which access by these users to applications and other services information is granted or denied*" [2]. In practice, an access control mechanism is a

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 274–288, 2013.

system that enforces an access control policy [3], and can be seen as a competitive advantage for organizations.

Several works exist on the definition of methodologies and tools to enforce access control policy, such as RBAC (*Role-Based Access Control*) [4], its extension P-RBAC (*Privacy-Aware Role-Based Access Control*) [5], XACML (*eXtensible Access Control Markup Language*) [6], among others. However, users tend to avoid using them due to its complexity, which makes the access control policies design a difficult and error prone task [7][8].

This paper proposes a simple, flexible and scalable policy model for access control and data privacy protection. The idea is that different policies can be derived from the model in a simple manner and later enforced through a protection layer. The model allows users to define the policies without requiring specific or in-depth knowledge about the web application. Consequently, it reduces the organization dependence on the development teams to decide which information can be accessed, how, and when. A prototype tool was implemented and an experimental evaluation was conducted to study the effectiveness of the proposed approach, both in terms of performance and scalability. Results show that, in practice, the proposed approach allows enforcing access control in a non-intrusive way and with a minimal performance overhead. This suggests that the mechanism can be used as a simple and effective alternative to privacy protection, avoiding using complex and costly solutions.

The paper is organized as follow. Section 2 introduces background concepts and related work. Section 3 presents the proposed approach, including the requirements and characteristics needed for building a good access control mechanism. Section 4 presents the experimental evaluation and a comparison between the proposed approach and existing techniques in terms of key quality attributes and features. Finally, Section 5 presents the conclusions and future work.

2 Background and Related Work

To protect data and resources against unauthorized disclosure and unauthorized or improper modifications, an access control process is required. This process is usually based on three key concepts: the **security policy** (the high-level rules according to which access control must be regulated), the **security model** (formal representation of the access control security policy and its working), and the **security mechanism** (the low level (software and hardware) functions that implement the controls imposed by the policy and formally stated in the model) [9].

Various works and technologies have been proposed to support access control, including approaches based on privacy policies. The problem is that, in most cases, existing approaches are limited due to their incompleteness or because they are not scalable or portable enough [10-12]. Also, such approaches are normally very complex and frequently lead to increased costs in terms of application development and maintenance, creating the need for simpler and less expensive solutions. In the next paragraphs we briefly survey the most relevant technologies for our research and

point out the main differences among those technologies and our solution (a comparison based on quality attributes is presented in Section 4.1).

P3P (*Platform for Privacy Preferences Project*) [13] is a protocol that allows websites to declare, in a standard format, the intended use of the information they collect about users, such as *what* data is collected, *who* can access those data and *for what* purposes, and for *how long* the data will be stored. This information can be retrieved automatically and is easily interpreted. Even though the P3P protocol provides a standard mean for enterprises to define privacy promises to their users, it does not provide any mechanism to ensure that these promises are consistent with the internal data processing, i.e. it does not address the issue of how efficient the web applications are on enforcing these policies when accessing the data.

To complement P3P's lack of enforcement mechanisms, many privacy-aware access control models have been investigated. Byun and Li [14] proposed a privacy preserving access control model based on P3P *purposes* (elements that defined the intended use of data) for relational databases. Accesses are granted if the access purpose defined by the requestor is among the allowed ones (previously defined by the user). Nevertheless, access decisions based only in the P3P purpose element are frequently insufficient and more rules and elements are needed to describe disclosure decisions. In this context, Agrawal et al. [15][16] proposed a framework for fine-grained access control implementation in a Hippocratic database system. It provides column level, row level, and element level access control. However, as it is based on Hippocratic database systems, the applicability of the approach is quite limited.

The Role-Based Access Control (RBAC) [4] is a well-known model for organizational access control, where permissions are assigned to roles and roles are assigned to users. The P-RBAC (*Privacy-Aware Role-Based Access Control*) [5] is an extension of RBAC that incorporates notions of privacy policies. These policies are based on rules that need to be carried out by the organization after access is granted to the user [17]. Although quite powerful, P-RBAC has some limitations, including: need for algorithms for detecting redundancy in conditions; limited support of only one type of relation (referred to as AND) among different permission assignments [18] [19].

The XACML (*eXtensible Access Control Markup Language*) [6] is a standard that describes both a policy language and an access control decision request/response language (both written in XML). The policy language is used to describe general access control requirements, and the request/response language is used to query a decision engine that evaluates access requests against existing policies. However, the request/response decisions are specific constructions that limit the use to web applications adopting XACML. Thus, it cannot be used in a generic manner, but only in the context of applications that implement these constructions (i.e. applications have to be specifically designed with XACML).

In this work we define a model from which one can easily derive access control policies. The model is simple, flexible and scalable, allowing complex restrictions to be defined according to the security and privacy policies of the organization. To enforce the policies defined through the model, a protection layer was implemented as a security mechanism, which can be used with several database management systems.

The mechanism is in charge of monitoring the data access, guaranteeing access control and data privacy, with minimum performance impact. A detailed comparison (in terms of quality attributes and main features) of our approach with the techniques presented above is presented in Section 4.1.

3 The Proposed Approach

The purpose of access control policies is to restrict unauthorized users from accessing unauthorized data and, consequently, to protect data privacy. Our approach is based on a model that uses a XML-schema language to support the definition of policies and on an independent layer that, according to the existing policies, filters the responses sent from the database to the client web application, permitting or denying the information to the user who requests them. Figure 1 illustrates the process. As shown in the example, a user requests some information from the database through a web application (step 1). This request arrives at the database and all the information is recovered (step 2). Then, the security layer processes the database response according to the predefined access control policies and only the permitted information is delivered to the user (step 3). Obviously, there are many options for implementing this security layer. It can be implemented as a mechanism in the database, as a proxy, or be integrated into the web application using the Aspect-Oriented Programming (AOP) paradigm. As an example, in Figure 1 we represented it as a proxy.

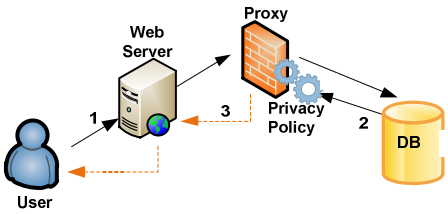


Fig. 1. The proposed mechanism and steps to apply privacy policy

3.1 Needs and Requirements for Access Control

To drive the design of the solution, we conducted a study to identify the real needs that should be tackled by a policy model and the functional and non-functional requirements relevant for implementing a good access control mechanism. This study included two steps: 1) analysis of existing bibliography on the characteristics of access control mechanisms, and 2) interviews with IT professionals interested in privacy and access control to better understand the real demands from the field.

While observing the needs of healthcare enterprises, Beznosov [20] identified some factors that should be used to support authorization decisions in order to comply with patient information discloser requirements. These factors are related to *roles* and *conditions*. *Roles* are groups of profiles to which users are assigned during a given session. The *conditions* should be expressed in terms of *location* (from where the user

is accessing information) and *time* (the period of time the user is accessing information). Conditions should be satisfied before an access request can be granted.

Bertino et al [21] identified the requirements that must be satisfied by an access control system for XML documents. According to the authors, the access control policies should basically define *who* can have access, *what* can be accessed, and *under which conditions* can certain actions be executed. Authors also mention the importance of roles, flexibility (to support one or many policies), time (to constraint the access to specific temporal intervals), and support for both browsing (allow subjects to see information) and authoring (allow subjects to modify information) policies.

Lu et al. [22] and Tolone et al. [23] presented some requirements for access control in collaborative systems. Among them, the most general are: transparency and flexibility (the access control model must facilitate transparent access for authorized users and exclusion of unauthorized users in a flexible manner), simplicity (the effects produced by access control on the system must be clear and easy to understand), expressiveness (the access control model must be expressive enough to specify complex access policies at different levels of detail), and scalability (the access control model must be able to support a rich number of operations).

Based on the literature, Vimercati et al [24] discussed the main desiderata for access control systems and illustrated the main characteristics of access control solutions. These can be summarized as follows: (i) be expressive, flexible, and simple in terms of use and implementation; (ii) Allow completing and correcting the policy specification, implementing *conditions*; (iii) Support groups (e.g. employees, programmers, consultants); (iv) support positive and negative policies; (iv) allow attribute-based specifications (access restrictions to the data/services should be expressed by policies that specify the properties (attributes) that a requester should have to gain access to the data/services).

As mentioned before, to support the design of the proposed policy model, several IT professionals were interviewed to understand their expectations regarding the policy definitions. This process was conducted as follows: a database system representing a network of book stores that sell books in physical stores and in an online system was carefully designed and described with a high level of detail (available at [25]). Four profiles were defined to access the system: customer (online and physical stores), seller (physical stores), manager (one manager for each physical store), and administrator (online and physical stores). Also, information about the time intervals and the location from where each role can access data was provided. The system description was emailed and each interviewee was requested to define the policies he deems of interest to control the data access. The goal was to allow interviewees to freely describe which information they think are important to be preserved and in what conditions that information should be preserved according to the existing user profiles.

Initially, 30 professionals were contacted. They were selected based on their insertion in the IT market, considering aspects like being working in big companies, in positions where privacy concerns are mandatory. From the initial 30 professionals, 22 answered indeed.

The responses were analyzed and the major concerns of the respondents were about the data each profile can access, with some notes about the conditions (as time period and location from where the access is done). To better understand the context, the following points present examples of the responses gathered for each role:

- **Administrator:** must have access to all tables in the database, for data inclusion, deletion and modification;
- **Seller:** must have read-only access to registered items (table *item, author*). He can include new orders in the database and read only his own orders (table *orders, order_line, cc_xacts*). He can include, delete or modify customers' data (table *costumer, address* and *country*).
- **Customer:** to be registered in the system, a customer must have access to write his data (table *costumer, address* and *country*) and can access or modify only his own data. Must have read-only access to registered items (table *item, author*) and his own orders (table *orders, order_line, cc_xacts*). He cannot have access to some specific information, as the cost of the item or the discount in the order (field *i_cost* and *ol_discount).*
- **Manager:** must have access to all tables in the database, for data insertion and modification;

Approximately 42% of the respondents included aspects related to the time period and the location from which the different profiles could access the system. For example, they stated that the sellers should access the system only through local systems in physical stores and during the operating period: from Monday to Friday from 8:00 a.m. to 6:00 p.m. and on Saturdays from 9:00 a.m. to 1:00 p.m. Also, customers should have online access 24 hours a day.

From this information it is possible to identify the need for policies specifications that, first, restrict columns and rows from tables considering different profiles, groups or roles. Even working in different positions, all of the IT professionals agree this is the most important requirement to help maintaining data privacy. The policy specifications also should restrict the time period and the location from which the access is performed (almost half of the interviewed described this as an important feature). Aligned with the analysis of the existing bibliography performed in the first step of this work, this research reinforces the importance of using conditions and groups to define policies, as well as its expressiveness.

Based on the two steps presented above, the requirements for the policy model and the access control mechanism were defined as (these are discussed and verified in more detail in Section 4.1):

- **Functional** (related to the specific features that should be implemented): the approach should support the correct and complete implementation of (i) conditions; (ii) groups; (iii) positive or negative policies.
- **Non-Functional** (regarding quality attributes of the model and mechanism): the mechanism should be design in a way that guarantees: (i) portability across different web applications and databases; (ii) non-intrusiveness in terms of the web application and database design and functioning; (iii) low performance impact in the

processing of the database responses; (iv) scalability to databases of different sizes, with increasing numbers of users and policies; (iv) simplicity in terms of policies definition and mechanism usage.

3.2 Policy Model

The policy model is represented in an XML-schema file that contains the definition of the structure of an XML document and is used to derive different policies from the same structure. As illustrated in Figure 2, the model implements all the characteristics mentioned in the previous section. In a broad view, it defines *who* can access certain information, *when, from where, and how* the required information can be accessed.

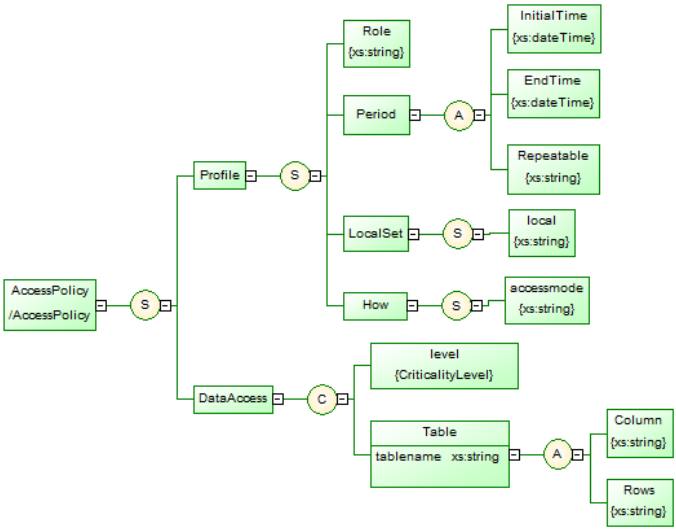


Fig. 2. Policy model

The tag *AccessPolicy* in the model is the start point to define a policy. It is composed by two other tags: the *Profile* and *DataAccess*, both connected by a selector *S*. The selectors *A, S* and *C* mean, respectively, *All, Sequence* and *Choice*. *Sequence* means that all the elements must appear at least once in the order of their declaration. *Choice* means that one element must be chosen. And *All* means that each element can appear or not, in any order [26].

The *DataAccess* offshoot was designed to meet the requirements of restricting columns and rows information from tables, which is the major concern of IT professionals interviewed during the development of this work. From *DataAcess*, two tags were created and must be chosen: *Level* or *Table*. The *Level* tag represents an alternative way of restricting the information: according to their criticality level. This tag is foreseen for the model, but its implementation is thought for future work. The *Table*, *Columns* and *Rows* tags represent the information in table columns and rows that will be restricted.

The *Profile* offshoot was designed to implement the groups and conditions requirements, described as very important in the literature and also pointed as necessary by almost half of IT professionals interviewed to this work. The *Role* tag represents the user groups. The *Period* tag restrict when the information can be accessed in terms of date and hour (through *InitialTime, EndTime* and *Repeatable* tags). *LocalSet* tag restricts one or more locations from which user access the information (e.g. local network or remote access). *How* tag represents the means through user can access information (e.g. from a web application, web service or a SQL console).

From the model presented in Figure 1, it is very simple to derive policy files, which are represented as XML files.

3.3 Access Control Mechanism

Access policies defined based on the model previously described are applied in practice by an access control mechanism. For each data access, this mechanism identifies the applicable policies and performs the required data filtering. In practice, the user requests normally access the database. Then, all the requested information is recovered from the database, but it filtered in the application before being returned to the client application (i.e. to the user). When returning the information, the mechanism verifies and masks columns and rows information, denying/granting access to the information according to the applicable policies.

There are several options to implement the access control mechanism, including: (i) r**elational database systems extensions** (database Management Systems (DBMS) can be modified to implement the privacy policies, using for instance, triggers, views or language constructs); (ii) u**se of a proxy** (during the request/response process the proxy is able to intercept for reading or modifying both the request and response, allowing or denying information according to the policies); (iii) **application instrumentation** (the client application can be instrumented to intercept the database requests). As the purpose of the access control mechanism is to be easy-to-use, application instrumentation was the approach chosen to implement the prototype used in this work. In practice, we use AOP to intercept key web application execution points and mask the information not permitted. Although it has the limitation of being dependent of the application, this option is easy to implement and very useful to develop prototypes and evaluation systems. The mechanism was developed in Java and supports the definition of policies according to the proposed model. To use it, is necessary to create the policies in a specified folder and add the driver file to the application server, replacing the JDBC driver previously in use.

3.4 Analysis of the Characteristics of the Access Control Mechanism

According to the requirements defined in Section 3.1, the proposed solution has the following characteristics (see also Table 1, last column):

- It implements *Conditions*, represented by the tags *Period*, *Localset* and *How; and is* based on groups, represented by the tag *Roles;*

- It supports positive policies. We decided to implement only the positive ones as using both types of policies would require some rule combination algorithms, which would compromise one of the main goal of the solution: simplicity;
- It provides high correctness and completeness. The structure allows finer granular access control, combining columns and rows, than simply denying or granting access to specific information;
- The policies are defined trough XML files, which is a portable technology. The model permits the implementation in several languages (C#, .Net, Java). The prototype used in this work is implemented in Java, which is also a portable language across many platforms;
- The implemented prototype is non-intrusive. In fact, the mechanism intercepts the return of the query to filter the permitted information, so, it is not intrusive to the database. Also it uses the AOP concepts, which can be used with the minimal interference in the web application or web service;
- It presents low performance impact. Experiments were performed to evaluate the performance of the proposed solution and results showed that the performance impact is low and acceptable (it will be presented in section 4);
- It is scalable, as it allows adding new policies through the XML files and, according to the performed experiments, the number of policies does not affect the performance (will be also presented in section 4);
- Its main characteristic is simplicity. Comparing to the other technologies cited in this paper, the simplicity is a great advantage of the proposed solution. Due to the simplicity of the model and the XML technology, the model (i) permits that policies specifications can be kept under control; (ii) requires less specific knowledge from the person responsible for specifying the policies; (iii) permits easy integration with existing technology.

4 Evaluation

This section presents a comparative study between our solution and existing technologies (according to the predefined requirements) and an experimental evaluation to verify the scalability and performance of our solution. The goal is to better understand the potential of the model and of the mechanism proposed, and to demonstrate the advantages and disadvantages of using them.

4.1 Comparison with Existing Technologies

To better understand the potential of the proposed solution, Table 1 shows a comparison summary considering different technologies (according to the requirements presented in Section 3.1). The last column, labeled as “proposed solution”, represents our approach. Implementation levels were defined to classify the technologies according to the requirements. For the functional requirements, the levels were defined as *implemented (I), partially implemented (P)* and *not implemented (N). Partially implemented* means that only part of the requirement is implemented. For example, to

the *negative and positive policies* requirement, technologies classified with *P* uses only one type of policy (negative or positive). For the non-functional requirements, the implementation levels were defined as *high (H), medium (M)* and *low (L).*

Table 1. Comparison of the requirements in relation to different technologies

	Requirements	P3P	FGAC	RBAC	XACML	PROPOSED SOLUTION
Functional	Conditions	N	I	P	I	I
	Groups	N	I	I	I	I
	Negative and Positive Policies	N	P	P	I	P
	Correctness/ Completeness	P	I	I	I	I
Non-functional	Portability	H	M	M	H	H
	Non-intrusiveness	H	H	H	H	H
	performance impact	L	M	L	M	L
	Scalability	H	H	H	M	H
	Flexibility	H	H	H	H	H
	Simplicity	H	L	M	L	H

Functional Requirements	
Implementation level	Description
I	Implemented
P	Partially implemented
N	Not Implemented

Non-Functional Requirements	
Implementation level	Description
H	High
M	Medium
L	Low

As we can observe, most of the functional requirements are not implemented in P3P. The reason is that P3P does not permit or deny information; it only presents a snapshot summary of how the site collects, handles and uses personal information about its visitors. Then, P3P-enabled Web browsers and other P3P applications will read and understand this snapshot information automatically, compare it to the Web user´s own set of privacy preferences, and inform the user when these preferences do not match the practices of the Web site he or she is visiting [13].

For the FGAC (Fine-Grained Access Control) [15], the medium implementation level of portability is due to the policies that are extensions of SQL language and is implemented into the DM5 database [27]. Some changes must be implemented when using other databases. The medium classification of the performance impact is due to the FGAC policy indeed affects the performance [27]. The simplicity is classified as *low* for two main reasons: (i) the users need to have advanced knowledge about queries and passing parameters, even object-oriented concepts to create instances of the policies; (ii) many policies to the same role, with many instances, can be difficult to manage and complex to keep under control.

According to Butler [12], the implementation of RBAC is different in every operating system. There is currently no standardization for implementation. This makes single console management and compliance difficult in most enterprises that administer heterogeneous systems [12]. That is the reason for the portability to be classified as medium.

The medium implementation level to the simplicity in RBAC is due to its many dimensions. According to Sandhu [28], it can be very complex embodying generalization and specialization hierarchies, such as found in object-oriented systems (although

widely used, is necessary to implement it considering concepts like role hierarchy, separation of duty, etc). For the XACML, the same requirement is classified as *low* because it is very verbose and extended conditions expressed in XACML quickly become unreadable [8]. The medium level for performance impact to XACML is because, according to Miseldine [29], the placement of data within an XACML policy will influence the performance of its interpretation.

For the proposed solution, the negative and positive policies requirement was classified as partially implemented because we decided to implement only the positive policies. It makes our solution slightly limited comparing to XACML, which implements both policy types. However, XACML uses rule combination algorithms, which contribute to its complexity (the *simplicity* requirement of XACML was classified as low). In face of several technologies with low and medium levels of simplicity, we tried to keep this feature as high as possible, being the main advantage to the proposed solution.

4.2 Scalability and Performance Evaluation

A case study was developed to evaluate the scalability and performance impact of the proposed approach. Scalability is expressed in terms of the number of policies being processed by the mechanism, i.e., the goal is to understand if the number of policies affects the performance. The performance impact can determine advantage or disadvantage of using the mechanism. For these experiments the performance was characterized by the average response time and throughput.

The web application used in the experiments is a Java implementation of a *TPC-W* [30], that is a benchmark for web-based transactional systems where several clients access the website to browse, search, and process orders. The database used in the experiments is Oracle [31] and the metrics were collected using the JMeter tool [32].

Tests were performed considering requests representing 4 different application use scenarios, with different user profiles. The first scenario represents requests of a registered user adding books to the basket and buying them using a valid login/password. The second scenario represents requests of an unknown (guest) user only seeing some books and making a search, but not adding items to the basket nor making any purchase. The third scenario represents requests of an unknown user selecting some items and having to register himself before making the purchase. Finally, the forth scenario represents requests of a registered user buying many books and seeing his last order. Different policies were implemented for the same profile, each one representing different restrictions across the model. They are available to all scenarios. The prototype filters all the corresponding policies of the user logged in the application to apply them. Independent tests were conducted to verify the correctness of policy implementation.

For each scenario was used, respectively, 5, 25, and 125 different policies files. The simulations of user (threads) ranged from 1 to 128 users for each set of policies. Also, in order to understand the performance impact, the tests were also performed without the access control mechanism in place (to obtain baseline indicators). The results are summarized in Figure 3, which is followed by the result discussion.

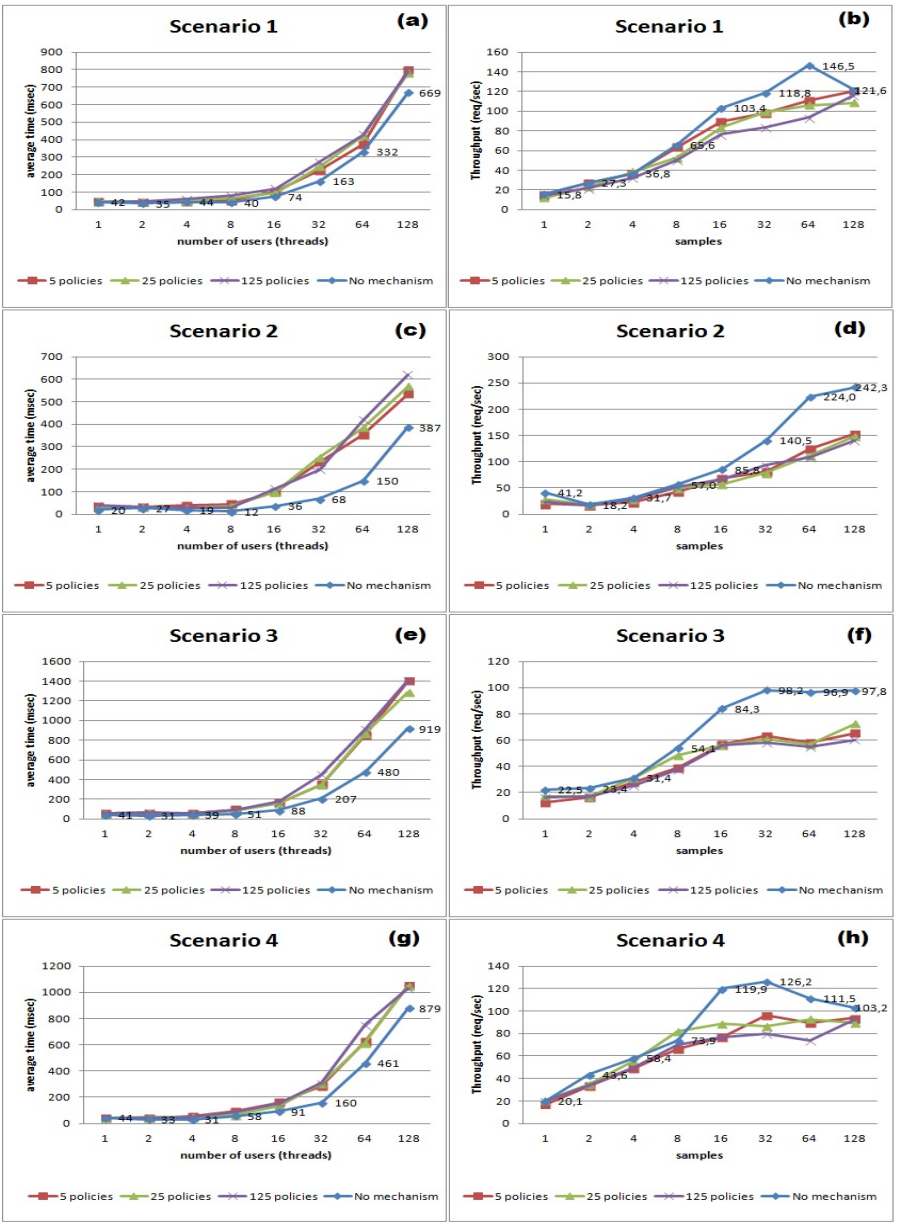


Fig. 3. Scenarios average time and throughput

Figure 3 shows the average processing time of all requests for each scenario. This average time is given in milliseconds and is presented in Figures 3a, 3c, 3e and 3g. Also, Figure 3 shows the throughput results. Throughput is calculated as requests divided by unit of time. The time is calculated from the start of the first sample to the end of the last sample, including any intervals between them, as it is supposed to

represent the load on the server. The throughput results are presented in Figures 3b, 3d, 3f and 3h

Through the analysis of the average processing time requests is possible to observe that, first, the proposed solution is highly scalable in terms of numbers of policies. The use of large number of policies can be necessary because of the different quantities of information in the databases (tables and rows) and different user profiles. In Figures 3a, 3c, 3e and 3g the number of policies barely affects the system performance. As the average time of experiments are in milliseconds, the small variation of values between the data representing 5, 25 and 125 policies, respectively, is totally acceptable.

Still analyzing the average processing time requests, now in terms of performance, Figures 3a, 3c, 3e and 3g shows that the proposed solution has very low impact when few users are using the web application. The average time without the access control mechanism is very similar to the other results until around 8 users. As the number of users increases, little differences arise. We found that, although the inclusion of the access control mechanism affects the performance for higher number of users, the system performance is rational, close to linear and acceptable.

The throughput variation presented in Figures 3b, 3d, 3f and 3h is similarly acceptable for the policy number variation, reinforcing the scalability of the solution. The samples are given by the number of users multiplied by the number of the requests of each scenario. The performance impact in the throughput analysis is also similar to average processing time results: about the 8 first samples the results with and without the mechanism are similar and the differences arise as it increases.

Figures 3b and 3h present a considerable decrease in the throughput variation with no access control mechanism to the last samples (about 64 samples in Figure 3b and 32 samples in Figure 3h respectively). It happens due to the requests the both scenarios have in common, that is the login and password request to buy the books. The JMeter tool possibly uses the same registered users to perform the tests and the user data (login and password) may be in cache or in cookies, making the process faster.

5 Conclusions and Future Works

Providing efficient and effective policies and an access control mechanism for assisting the privacy in web applications has long been an unresolved issue. However, it is critically important for Internet-based data management systems. Motivated by the advantages of the XML language, this study investigated a practical solution for this problem. We defined a model to describe the security policies and proposed a prototype tool to intercept information from the database, permitting or denying this information according to the policies derived from the model. In comparison with other important technologies, our solution has two important features. First, it allows creating policy instances from the policy model, which is flexible and extensible. This alleviates the teams that develop web applications and allows reducing errors. Second, the simplicity of the solution makes it practical, flexible, easy to implement and use, even being a intrusive and dependent of the application.

We implemented the policy model and the prototype tool in a TPC-W web application and tested the scalability and performance. The tests results are very promising. The scalability is high because the high number of policies did not affect significantly the performance. The performance impact, although it become greater as the number of users (threads) increases, can be considered quite acceptable front of the importance of protecting information.

As future work, we intend to complement the tests, evaluating the solution in large scale applications. Also, we intend to integrate this solution in a more complex environment, where attacks are identified and the information under attack is permitted or denied according to its critically level, classified previously. It helps to avoid false positive results from the attack detection tools. This is one of the areas that we are currently working on.

References

1. Bertino, E., Lin, D., Jiang, W.: A Survey of Quantification of Privacy Preserving Data Mining Algorithms. In: Aggarwal, C.C., Yu, P.S., Elmagarmid, A.K. (eds.) Privacy-Preserving Data Mining, vol. 34, pp. 183–205. Springer, US (2008)
2. Internet Engineering Task Force (IETF), http://www.ietf.org/ (accessed: September 07, 2012)
3. Bertino, E., Ghinita, G., Kamra, A.: Access Control for Databases: Concepts and Systems. Now Publishers Inc. (2011)
4. Sandhu, R.S.: Role-based Access Control. In: Advances in Computers, vol. 46, pp. 237–286. Elsevier (1998)
5. Ni, Q., Bertino, E., Lobo, J., Calo, S.B.: Privacy-Aware Role-Based Access Control. IEEE Security Privacy 7(4), 35–43 (2009)
6. OASIS eXtensible Access Control Markup Language (XACML) TC | OASIS, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml (accessed: September 07, 2012)
7. Masi, M., Pugliese, R., Tiezzi, F.: Formalisation and Implementation of the XACML Access Control Mechanism. In: Barthe, G., Livshits, B., Scandariato, R. (eds.) ESSoS 2012. LNCS, vol. 7159, pp. 60–74. Springer, Heidelberg (2012)
8. Bernard Stepien, S.M.: Advantages of a non-technical XACML notation in role-based models, pp. 193–200 (2011)
9. Samarati, P., de, S., di Vimercati, C.: Access Control: Policies, Models, and Mechanisms. In: Foundations of Security Analysis and Design (Tutorial Lectures), pp. 137–196 (2001)
10. Bernard Stepien, S.M.: Advantages of a non-technical XACML notation in role-based models, pp. 193–200 (2011)
11. Turkmen, F., Crispo, B.: Performance evaluation of XACML PDP implementations. In: Proceedings of the 2008 ACM Workshop on Secure Web Services, New York, NY, USA, pp. 37–44 (2008)
12. Michael Butler, J.: Extending Role Based Access Control - A SANS Whitepaper, http://www.sans.org/reading_room/analysts_program/access-control-foxt.pdf (accessed: February15, 2013)
13. P3P: The Platform for Privacy Preferences, http://www.w3.org/P3P/ (accessed: September 04, 2012)

14. Byun, J.-W., Li, N.: Purpose based access control for privacy protection in relational database systems. The VLDB Journal 17(4), 603–619 (2008)
15. Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S., Rjaibi, W.: Extending Relational Database Systems to Automatically Enforce Privacy Policies. In: Proceedings of the 21st International Conference on Data Engineering, Washington, DC, USA, pp. 1013–1022 (2005)
16. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: 28th Int'l Conference on Very Large Databases, Hong Kong (2002)
17. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer 29(2), 38–47 (1996)
18. Arora, S., Song, E., Kim, Y.: *Modified* hierarchical privacy-aware role based access control model. In: Proceedings of the 2012 ACM Research in Applied Computation Symposium, New York, NY, USA, pp. 344–347 (2012)
19. Ni, Q., Bertino, E.: Conditional Privacy-Aware Role Based Access Control. Springer, Heidelberg (2007)
20. Beznosov, K.: Requirements for access control: US Healthcare domain. In: Proceedings of the Third ACM workshop on Role-based access control, New York, NY, USA (1998)
21. Bertino, E., Carminati, B., Ferrari, E.: Access control for XML documents and data. Inf. Secur. Tech. Rep., vol. 9, n° 3, pp. 19–34 (July 2004)
22. Lu, Y., Zhang, L., Sun, J.: Task-activity based access control for process collaboration environments. Comput. Ind. 60(6), 403–415 (2009)
23. Tolone, W., Ahn, G.-J., Pai, T., Hong, S.-P.: Access control in collaborative systems. ACM Comput. Surv. 37(1), 29–41 (2005)
24. De Capitani di Vimercati, S., Samarati, P., Jajodia, S.: Policies, models, and languages for access control. In: Databases in Networked Information Systems, pp. 225–237 (2005)
25. Regina Lúcia de Oliveira Moraes, `http://www.ft.unicamp.br/~regina/Gerais/Request-database-administrator-detailed.pdf` (Accessed: April 9, 2013)
26. Sybase XML Modeling PowerDesigner® 15.3, `http://wwwdownload.sybase.com/pdfdocs/pdd1100e/xmug.pdf` (accessed: April 09, 2013])
27. Zhu, H., Lü, K.: Fine-grained access control for database management systems. In: Proceedings of the 24th British National Conference on Databases, Berlin, Heidelberg, pp. 215–223 (2007)
28. ROLE-BASED ACCESS CONTROL A Position Statement, `http://profsandhu.com/misc_pubs/nist/n94rbac.pdf` (accessed: January 29, 2013)
29. Miseldine, P.L.: Automated XACML Policy Reconfiguration for Evaluation Otimisation. In: Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems (SESS 2008), pp. 1–8. ACM, New York (2008)
30. TPC-W, `http://www.tpc.org/tpcw/` (accessed: January 08, 2013)
31. Oracle | Hardware and Software, Engineered to Work Together, `http://www.oracle.com/index.html`(accessed: January 29, 2013)
32. Apache JMeter - Apache JMeter™, `http://jmeter.apache.org/` (accessed: January 09, 2013)

Behavioral Tendency Obfuscation Framework for Personalization Services

Ryo Furukawa, Takao Takenouchi, and Takuya Mori

Cloud System Research Laboratories, NEC Corporation, Kanagawa, Japan
r-furukawa@cb.jp.nec.com, takenouchi@bu.jp.nec.com,
moritaku@bx.jp.nec.com

Abstract. Web service providers collect user behaviors, such as purchases or locations, and use this information to provide personalized content. While no provider can collect behavioral information across different service providers, the behaviors for all service providers are accumulated in a user's terminal. If a provider could analyze these behaviors stored in the terminal, it could provide more valuable services to the user. There is a problem, however, in that sensitive user information would be revealed when the provider obtained behaviors related to other services. This sensitive information consists of the user's behaviors and characteristic tendencies analyzed from the collected information. In this paper, we propose a model for preserving privacy, called *ρ-tendency certainty*, which considers breaches of privacy from collected information. We also propose a behavioral tendency obfuscation framework, which sends dummy queries to service providers in order to satisfy *ρ-tendency certainty*. Experimental results show that the proposed framework can satisfy *ρ-tendency certainty* with a few number of dummy queries and create dummies within 1 msec, thus the proposed framework is applicable to real services.

Keywords: privacy, personalization services, behavioral tendency obfuscation framework.

1 Introduction

Recently, more and more service providers collect information about user behaviors, such as purchases, TV or video on demand (VOD) programs viewed, and locations obtained from GPS information. Such services also provide personalized content that related to these collected user behaviors. For example, Amazon collects user information about buying or browsing merchandise in order to recommend merchandise that might be of interest. Similarly, Google provides behaviorally targeted advertisements by collecting and analyzing the search words and histories of users browsing websites. Users can thus receive personalized content from each of these services.

While each service provider cannot collect a user's behaviors across different service providers, the behaviors with respect to many service providers are accumulated in the user's terminal, such as a personal computer or smart phone.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 289–303, 2013.

That is, information about purchases on Amazon and web searches on Google is accumulated in the same terminal. Thus, if a service provider could obtain a user's behaviors from the terminal, they could provide more accurate, valuable personalized content to the user. For example, if one e-commerce service could use information about the TV programs watched by a user, it could then recommend merchandise introduced in these programs or related to actors appearing on them.

There is a problem, however, in that sensitive user information can be revealed when a service provider obtains behaviors related to other services. For example, the user does not want the e-commerce service know behaviors about TV programs watched by user, because it may contains secret TV programs. Hence, it is necessary to modify such user behaviors to preserve privacy at the terminal.

Many technologies have been proposed to solve this problem, such as perturbation techniques[1]. This type of technique protects sensitive information by adding noise to information at the user terminal, while preserving the statistical nature of the aggregated user information. This prevents analysis of individual user data for providing personalized content.

In the research area of document search, the TopPrev framework[2] takes the query obfuscation approach. In this approach, a client sends both an actual query and dummy queries to a service provider. Then, the client receives all corresponding responses from the provider and selects those responses related to the actual query. This technique can protect sensitive information while still enabling personalized content for the user.

This technology, however, only considers the breach of privacy from a one-time query. The TopPrev framework assumes that the client connects to a service through an anonymous network[10] or uses query log anonymization[9]. When this assumption is removed, breaches of privacy from multiple queries must be considered. If a provider analyzed behaviors aggregated from multiple queries, it could reveal the user's characteristic behavioral tendencies, such as a category of products with frequent purchases or a TV genre with frequent viewing.

In this paper, to address this problem, we propose a model and framework for preserving privacy. The paper presents two main contributions. First, we define a model for preserving privacy, called *ρ-tendency certainty*, which ensures that the rate of identifying actual tendencies from analysis of collected behaviors is less than some value ρ. Second, we propose a behavioral tendency obfuscation framework that takes the query obfuscation approach and can satisfy *ρ-tendency certainty* efficiently. Our framework selects dummy queries through a heuristic approach that emphasizes fake tendencies and hides actual tendencies. We experimentally evaluate this framework and show that it can reduce the number of dummy queries required to satisfy *ρ-tendency certainty*, in comparison with using randomly created dummy queries. The proposed framework can reduce the communication traffic between a user's terminal and service providers. Consequently, the proposed framework can protects privacy from user behaviors at the user terminal with reasonable costs.

The rest of this paper is organized as follows. Section 2 represents the related work. Section 3 clarifies the problem definition. Sections 4 and 5 respectively introduce our new model, *ρ-tendency certainty*, and our proposed framework. Section 6 presents and discusses our experimental results, finally section 7 concludes the paper.

2 Related Work

Many researches have been proposed, that are useful for preserving privacy from the sensitive information at user terminals.

PIR (Private Information Retrieval)[3][4] enables information retrieval without leakage of retrieved items. In PIR, since a client encodes retrieval keys and inputs them to a server, retrieval keys are protected in term of computationally or information theoretically secure. The server sends all data in a database to the client with compressing these data based on the inputted key and the client obtains the result from it. PIR requires scanning and sending all data in the database, the computational and communicational costs are very high.

Some approaches have been proposed that retrieve information in more relaxed conditions. Pang[2] has proposed TopPrev framework that preserve privacy in the text search queries by obfuscating it. This framework creates dummy queries to obfuscate the queries considering change of probability distributions of topical intention occurred by the search queries, and selects the result related with a user's actual query. TopPrev framework can preserve privacy in the queries and search information with reasonable costs, but it do not assume that it applies to multiple query environment because it only considers the privacy of the one-time query. Kido[5] and Lu[6] have proposed approaches that are similar to Pang[2] for location based services (LBS). These technologies create dummy location information in order to hide the user's true positions. In their dummy creation algorithms, Kido[5] considers the moves of dummy positions and Lu[6] considers the size of convex hull of all positions in order to reduce a risk that observers may find true position. While these technologies can preserve the user's true positions, they also do not consider the user's behavioral tendencies.

On the other hand, cryptographic techniques such as searching on encrypted data[7] or PPDM (Privacy Preserving Data Mining)[8] have been proposed. These techniques can perform keyword-matching or data mining over encrypted data, thus the user's privacy is completely preserved. However each computations over encrypted data are computationally expensive.

3 Problem Definition

Let B be the universal set of information about a behavior and $b_t^a \in B$ be information about the user's actual behavior at time t. The value of t is incremented when the user's behavior is generated and not synchronized with another user's time stamp.

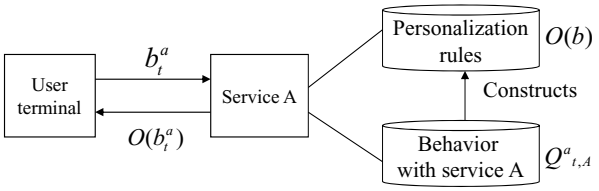


Fig. 1. Assumptions of a user's interaction with a service at time t

We focus on the kind of personalized service that returns content $O(b)$ when the user inputs behavior $b \in B$ as a query to the service. We assume that each service provider can only use the behaviors that it collects in constructing personalization rules, $O(b)$. For example, an e-commerce service constructs personalization rules by using only behaviors about merchandise bought through this service. The e-commerce service can recommend information about merchandise according to these personalization rules and inputted user behavior about merchandise bought through other e-commerce services. Let $Q_t^a = (b_1^a, b_2^a, ..., b_t^a)$ be the accumulated user's actual behaviors in the user's terminal at time t, and let $Q_{t,A}^a \subseteq Q_t^a$ be the user's actual behaviors in using service A. Fig.1 illustrates our assumptions of the user's interaction with service A at time t.

With these assumptions, two problems arise in regard to a user's privacy leakage, as follows.

Definition 1. *(Problem 1: Leakage of input information): If query b_t^a satisfies $b_t^a \notin Q_{t,A}^a$, then b_t^a is new information about the user's behavior with service A. Leakage of input information occurs if query b_t^a is inputted to service A and $b_t^a \notin Q_{t,A}^a$, and the service A can understand it is the user's actual behavior.*

Definition 2. *(Problem 2: Leakage of characteristic behavioral tendencies): Let Q_t^s be a user's behaviors collected by a service at time t. In general, service providers can estimate the user's characteristic behavioral tendencies from the collected behaviors Q_t^s, and if these are similar to actual behavioral tendency the user's privacy is breached. Let $S(Q_t^a, Q_t^s)$ be some measure of similarity between two sets of characteristic behavioral tendencies estimated from the user's actual behaviors Q_t^a and collected behaviors Q_t^s at time t. If $S(Q_t^a, Q_t^s)$ is high, then leakage of characteristic behavioral tendencies occurs.*

We regard a similarity function $S(Q_t^a, Q_t^s)$ as high, for example, when the value of $S(Q_t^a, Q_t^s)$ is higher than a value given by a user as a privacy requirement.

In addition to solving these problems, we must maintain quality of service at the same time. We can resolve problem 1 by using the query obfuscation approach adopted in [2]. This approach can also be applied to accurately receive valuable personalized content from a service. Hence, we next describe both this approach and problem 2 in more detail.

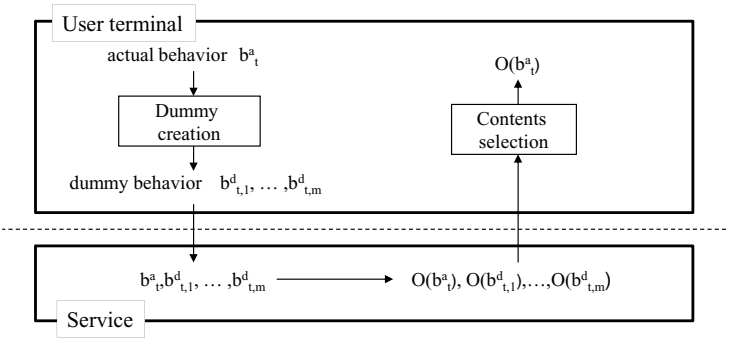


Fig. 2. Query obfuscation approach

3.1 Query Obfuscation Approach

The query obfuscation approach is a method that preserves the privacy of queries inputted to a service by obfuscating actual queries. Fig.2 shows the procedure of this approach. It involves a "dummy creation" function and a "content selection" function, which operate as follows. First, the dummy creation function creates dummy behaviors $b^d_{t,1}, b^d_{t,2}, ..., b^d_{t,m}$, where $b^d_{t,i}$ is a dummy behavior created at time t, and inputs $b^a_t, b^d_{t,1}, b^d_{t,2}, ..., b^d_{t,m}$ to the service when the user terminal inputs the actual behavior b^a_t. Second, the content selection function receives $O(b^a_t), O(b^d_{t,1}), ..., O(b^d_{t,m})$ and selects $O(b^a_t)$. In the system model, the service provider only can respond to one user's request $\forall b \in \{b^q_t, b^d_{t,1}, b^d_{t,2}, ..., b^d_{t,m}\}$ at one time and the user terminal serially sends a request b, hence the response $O(b)$ corresponds to the request b respectively and the user terminal can identify $O(b^a_t)$ from $O(b^a_t), O(b^d_{t,1}), ..., O(b^d_{t,m})$. Thus, this approach can obfuscate the actual behavior b^a_t and accurately receive content $O(b^a_t)$ related to the user's actual behavior.

In terms of communicational costs between the user terminal and the service, the optimal obfuscation is required introducing the least dummy queries in order to preserve privacy.

3.2 Limitations of Query Obfuscation Approach

While the query obfuscation approach can resolve problem 1 and maintain the quality of personalized content, it does not resolve problem 2, i.e., leakage of characteristic behavioral tendencies. Here, we assume that behavior $b \in B$ belongs to certain categories representing behavioral characteristics. Let C be the universal set of categories, and let $categories(b) \subset C \ for \ \forall b \in B$ be the categories of b.

Figures 3 and 4 show an example of discrete probability distributions $Pr(c|Q)$, $\forall c \in C$ of the frequencies for 20 categories in a set of user behaviors Q. Fig.3 shows the probability distribution $Pr(c|Q^a_t)$ constructed from the actual

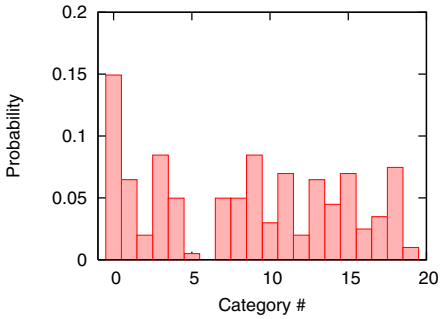


Fig. 3. Actual behavioral tendencies

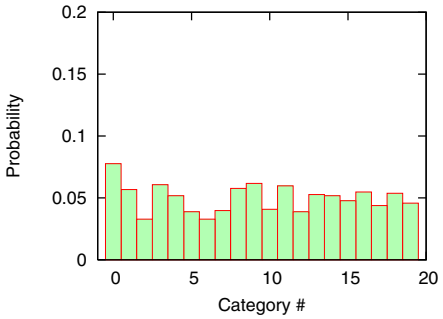


Fig. 4. Random behavioral tendencies

user behaviors Q_t^a, while Fig.4 shows the probability distribution $Pr(c|Q_t^s)$ constructed from behaviors Q_t^s containing randomly created dummy behaviors.

These figures show that the two rankings of categories ordered by probability in terms of $Pr(c|Q_t^a)$ or $Pr(c|Q_t^s)$ correspond. Specifically, categories 0, 3, 9, and 11 are contained in the top five categories in both Fig.3 and Fig.4. This shows that if dummy behaviors are randomly created, then some similarity measure between $Pr(c|Q_t^a)$ and $Pr(c|Q_t^s)$, that is, $S(Q_t^a, Q_t^s)$, becomes high. Thus, a random method cannot prevent leakage of characteristic behavioral tendencies. To solve this problem, we next devise a more effective dummy creation function for the query obfuscation approach.

4 ρ-*Tendency Certainty*

In this section, to address the privacy issue defined as problem 2, we propose a model for preserving privacy in collected behaviors. To preserve privacy, the query obfuscation approach must create dummy queries in such a way that collected behaviors satisfy this model.

Existing research, such as research on *l-diversity*[11] and it's derivations[12][13], defines models for preserving privacy by considering identification of a user's sensitive information. The *l-diversity* model focuses on the possibility that adversaries can associate sensitive information with a user identified from background knowledge. We thus designed our model for preserving privacy by analogy with such research.

In this paper, we assume that sensitive information in collected data is a characteristic category of a user's actual behaviors. This sensitive information represents the user's interests, and the problem is that this information can be identified by services.

Therefore, we have designed a model called *ρ-tendency certainty* by considering this possibility of identifying a user's actual interests from the behaviors collected by a service. First, we define new notions of the *actual tendency* and *service-side tendency*. Here, the *actual tendency* means the set of categories that

the user is actually interested in, while the *service-side tendency* means the set of categories estimated by the service to be the user's interests.

Definition 3. *(Actual tendency:) Given a user's actual behaviors Q_t^a at time t, let $IS(Q) \subset C$ be the set of categories estimated from some set of behaviors, Q. We denote the actual tendency at t as $IS(Q_t^a) \subset C$.*

Definition 4. *(Service-side tendency:) Given the behaviors collected by a service, Q_t^s, at time t, we denote the service-side tendency at t as $IS(Q_t^s) \subset C$.*

Next, we define the notion of *tendency certainty* in terms of the *actual tendency* and *service-side tendency*. This measure gives the probability that a service can identify a user's actual interests from collected behaviors.

Definition 5. *(Tendency certainty:) Given the behaviors Q_t^a and Q_t^s at time t, we denote the interest identification probability at t as $S(Q_t^a, Q_t^s)$, defined by*

$$S(Q_t^a, Q_t^s) = |IS(Q_t^a) \cap IS(Q_t^s)|/|IS(Q_t^a)|. \quad (1)$$

Here, the *tendency certainty* corresponds to the similarity described in the definition of problem 2. Thus, to address the privacy issue in problem 2, the *tendency certainty* should be low. Finally, we specify a *ρ-tendency certainty* model for preserving privacy in collected behaviors.

Definition 6. *(ρ-tendency certainty:) Given a user's privacy requirement ρ and behaviors Q_t^a and Q_t^s, the behaviors collected by a service, Q_t^s, satisfy ρ-tendency certainty if and only if $S(Q_t^a, Q_t^s) \leq \rho$.*

5 Proposed Framework and Dummy Selection Method

In this section, we propose a behavioral tendency obfuscation framework that satisfies *ρ-tendency certainty* by using the query obfuscation approach, and a heuristic dummy selection method.

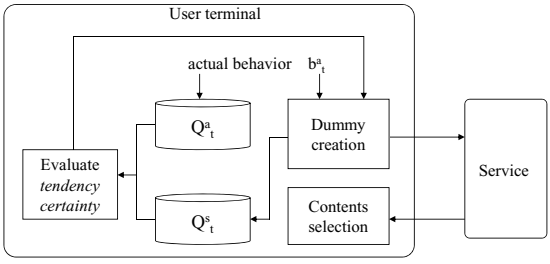


Fig. 5. Behavioral tendency obfuscation framework

Algorithm 1. Behavioral Tendency Obfuscation Framework

Require: behaviors Q^a_{t-1}, Q^s_{t-1}, user's actual behavior b^a_t.
1: $q_t = \{b^a_t\}$.
2: $i = 1$.
3: **repeat**
4: $\quad b^d_{t,i} \leftarrow DummySelection(Q^a_{t-1}, Q^s_{t-1}, q_t, b^a_t)$.
5: $\quad q_t \leftarrow q_t \cup \{b^d_{t,i}\}$.
6: $\quad i \leftarrow i + 1$.
7: **until** $\mathrm{S}(Q^a_{t-1} \cup \{b^a_t\}, Q^s_{t-1} \cup q_t) \leq \rho$.
8: $Q^a_t \leftarrow Q^a_{t-1} \cup \{b^a_t\}$.
9: $Q^s_t \leftarrow Q^s_{t-1} \cup q_t$.
10: send all behaviors in q_t to the service.
11: recieve $O(b)$ *for* $\forall b \in q_t$ and select $O(b^a_t)$.

5.1 Behavioral Tendency Obfuscation Framework

In the query obfuscation approach, a dummy creation function creates dummy behaviors, and both the actual behavior and the dummy behaviors are sent to a service. Thus, a user terminal can accumulate behaviors that are the same as those collected by the service.

Applying this characteristic of user terminals, we propose a behavioral tendency obfuscation framework. Fig.5 schematically illustrates this framework. The user terminal accumulates actual behaviors Q^a_t and behaviors inputted to services, Q^s_t. Hence, the user terminal can evaluate *tendency certainty* and create dummy behaviors by using only information accumulated in the terminal, without any information collected by services.

This framework is defined by Algorithm 1. In Algorithm 1, steps 3-10 are corresponding to procedures of the dummy creation function and step 11 is corresponding to procedures of the contents selection function. In this algorithm, when the user's actual behavior b^a_t at time t is inputted, the dummy selection function sequentially selects a dummy behavior. It does this by considering two behaviors, Q^a_{t-1} and Q^s_{t-1}, accumulated before time t, until ρ-*tendency certainty* is satisfied. Thus, when the user terminal sends selected queries q_t to the service, the collected behaviors Q^s_t satisfy ρ-*tendency certainty*.

5.2 Dummy Selection Method

In this section, we propose a dummy selection method for use in the proposed framework. The method selects a behavior $b^d_{t,i}$ from the universal set of behaviors, B. We next describe the requirements for the dummy selection method, before proposing a heuristic method satisfying these requirements.

Requirements for Dummy Selection Method. The dummy creation function must create a lower number of dummy queries in order to satisfy ρ-*tendency certainty*, because the network traffic between the user terminal and the service

Algorithm 2. Heuristic dummy selection method

Require: parameters $l, r, w1, w2$, universal set of behaviors B and categories C, behaviors Q^a_{t-1}, Q^s_{t-1}, q_t, b^a_t.

1: $Q^r \leftarrow RandomSetSelection(B, l)$.
2: $C^a_t \leftarrow IS(Q^a_{t-1} \cup b^a_t, r)$.
3: $P_{t-1}(c) \leftarrow Pr(c|Q^s_{t-1})$.
4: **for all** $b \in Q^r$ **do**
5: $\quad P'_t(c) \leftarrow Pr(c|Q^s_{t-1} \cup a_t \cup \{b\})$.
6: $\quad C^s_t \leftarrow IS(Q^s_{t-1} \cup q_t \cup b, r)$.
7: $\quad eval(b) = \sum_{c \in C^a_t}(P_{t-1}(c) - P'_t(c)) + w1 \sum_{c \in C^s_t \setminus C^a_t}(P'_t(c) - P_{t-1}(c)) + w2 \sum_{c \in C \setminus C^a_t \cup C^s_t}(P'_t(c) - P_{t-1}(c))$.
8: $b^o \leftarrow \underset{b}{\operatorname{argmax}}\, eval(b)$.
9: **return** b^o.

provider is proportional to the number of queries. To achieve this goal, we consider three requirements for dummy selection method that the method must satisfy, as follows:

1. The frequency $Pr(c|Q^s_t)$ of a category c in the *actual tendencies* should be decreased by dummy behaviors.
2. The frequency $Pr(c|Q^s_t)$ of a category c not in the *actual tendencies* should be increased by dummy behaviors.
3. Dummy behaviors should increase the frequency $Pr(c|Q^s_t)$ of a category c in the *service-side tendencies* but not the *actual tendencies* more than they increase the frequency of a category subject to requirement 2.

These requirements are designed to select dummy behaviors emphasizing categories in the *service-side tendencies* but not the *actual tendencies*. Therefore, such a dummy selection method can reduce *tendency certainty*.

Heuristic Dummy Selection Method. In this section we propose an efficient dummy selection method that operates heuristically and satisfies the above requirements. Practically, the size of universal set of information about a user's behavior $|B|$ is very large and determining the globally optimum dummy behaviors from $|B|$ is so hard, hence we adopt the heuristic method. Algorithm 2 defines the procedure of this heuristic dummy selection method.

This algorithm first randomly selects l behaviors Q^r, such that $|Q^r| = l$, from the universal set of behaviors, B, by using a function $RandomSetSelection(B, l)$. Second, the function $eval(b)$ is calculated for all behaviors $b \in Q^r$ to evaluate the inferences of these behaviors. Then, the behavior with the highest evaluation value b^o is selected. The function $eval(b)$ is defined as follows:

$$eval(b) = \sum_{c \in C^a_t}(P_{t-1}(c) - P'_t(c)) + w1 \sum_{c \in C^s_t \setminus C^a_t}(P'_t(c) - P_{t-1}(c)) + w2 \sum_{c \in C \setminus (C^a_t \cup C^s_t)}(P'_t(c) - P_{t-1}(c)). \quad (2)$$

The calculation of $eval(b)$ uses two sets of categories, C_t^a, C_t^s, related to the user's actual and simulated tendencies respectively derived from the function $IS(Q, r)$ and the probability of category $P_t(c)$ as derived by the function $Pr(c|Q)$. The function $IS(Q, r)$ derives r categories, that is, $IS(Q, r) \subset C, |IS(Q, r)| = r$, in which the user is interested from behaviors Q. This function is the parameterized version of the function $IS(Q)$ that used for the *actual* and *service-side tendencies*. The function $Pr(c|Q)$ derives the probability of $\forall c \in C$ as estimated from the behaviors Q. The implementations of the functions $IS(Q, r)$ and $Pr(c|Q)$ are described in the next section. The parameter $w1$ weights evaluation value for categories discribed in the requirement 3, and the parameter $w2$ weights evaluation value for categories described in the requirement 2. Because of requirement 3, $w1 > w2$ should be satisfied. The function $eval(q)$ outputs a higher value if dummy behavior b satisfies the requirements. Consequently, this heuristic dummy selection method also satisfies the requirements.

6 Experimental Evaluation

In this section, we describe an evaluation of the proposed framework through computer simulation. First, we evaluate the effectiveness of the heuristic dummy selection method in satisfying *ρ-tendency certainty*. Next, we evaluate the proposed framework in terms of the number of queries needed to satisfy *ρ-tendency certainty*.

We compare our approach with a random approach, because comparison with an existing method like the TopPrev framework[2] would be misleading. This method dose not assume that it applies to multiple query environment. Additionally, although general interests are assumed in the TopPrev framework, our approach does not require such information, and it is difficult to assume general interests for our target services.

6.1 Experimental Setup

Implementation of functions $Pr(c|Q_t)$ and $IS(Q_t, r)$. The function $Pr(c|Q_t)$ derives the occurrence probability of category c by calculating the frequency of behavior information in $Q_{[t-t_p,t]}$, as follows. Here, $Q_{[t-t_p,t]}$ means behavior information in Q_t during time span $[t - t_p, t]$, where Q_t is either Q_t^a or Q_t^s and the function $exist(b, c)$ return 1 for $c \in categories(b)$ and return 0 for the other:

$$Pr(c|Q_t) = (\sum_{b \in Q_{[t-t_p,t]}} exist(q, c)/|categories(b)|)/|Q_{[t-t_p,t]}| \quad (3)$$

The function $IS(Q_t, r)$ outputs r categories constructed from the top r categories c in the ranking given by $Pr(c|Q_t)$.

Algorithm 3. Modified framework

Require: behaviors Q^a_{t-1}, Q^s_{t-1}, user's actual behavior b^a_t, parameter k.
1: $q_t = \{b^a_t\}$.
2: $i = 1$.
3: **repeat**
4: $\quad b^d_{t,i} \leftarrow DummySelection(Q^a_{t-1}, Q^s_{t-1}, q_t, b^a_t)$.
5: $\quad q_t \leftarrow q_t \cup \{b^d_{t,i}\}$.
6: $\quad i \leftarrow i + 1$.
7: **until** $|q_t| > k$.
8: $Q^a_t \leftarrow Q^a_{t-1} \cup \{b^a_t\}$.
9: $Q^s_t \leftarrow Q^s_{t-1} \cup q_t$.
10: send all behaviors in q_t to the service.
11: recieve $O(b)$ for $\forall b \in q_t$ and select $O(b^a_t)$.

Program Architecture. We developed an experimental program that carries out the experiments as the agent simulation. In this program, the user agent has implicit behavioral tendencies, representing the occurrence probabilities of categories characterizing the user, and a behavior generator to generate an actual behavior b^a_t. The behavior generator randomly selects an actual user behavior from the universal set of behaviors, B, according to the implicit behavioral tendencies. The framework then receives b^a_t and creates a dummy query.

Dataset. We prepared universal sets of categories C and behaviors B by creating artificial data. The user's implicit behavioral tendencies were constructed by randomly setting a probability for each category $c \in C$. We also prepared one user agent for the experiments.

Environment. Each experiment was run on a virtual machine with a quad-cores processor and 4 GB of memory, running CentOS 5.6. The host server for this virtual machine had 12 cores consisting of 2.4-GHz processors and 196 GB of memory. The proposed framework was developed in Java 1.7.0_05.

6.2 Evaluation of Heuristic Dummy Selection Method

Here, we evaluate the heuristic dummy selection method by comparing with random approach. To better evaluate the heuristic dummy selection method, we modified the proposed framework to use the procedure of Algorithm 3. The modified framework adds dummy queries until the number of queries reaches k, instead of adding dummy queries until ρ-tendency certainty is satisfied. This modification enabled us to evaluate the proposed heuristic method under the condition of processing the same number of queries as the random method.

We also introduce the random dummy selection method. This method randomly selects behaviors from the universal behavior set B. Within the modified framework, $k-1$ dummy queries are selected randomly by this method and sent to the target service.

Table 1. Parameter settings

Parameter	Value	Explanation
$\|B\|$	200	Size of universal set of behaviors
$\|C\|$	20	Size of universal set of categories
r	5	Number of categories outputted by $IS(Q, r)$
k	5	Number of behaviors created in modified framework
l	20	Number of random behaviors in heuristic dummy selection
$w1$	0.2	Weight parameter in heuristic dummy selection
$w2$	0.1	Weight parameter in heuristic dummy selection
t_p	200	Time length used in $Pr(c\|Q)$
N	1000	Number of iterations

We make an experiment for comparing the heuristic dummy selection method and the random dummy selection method by using the modified framework. Table1 lists the experimental parameters for this evaluation.

First, we evaluated the *tendency certainty* of collected behavior sets created by the query obfuscation approach. Fig.6 compares the *tendency certainties* at each iteration between the heuristic and random dummy selection methods. In this figure, the *tendency certainty* for the heuristic method is consistently lower than that for the random method. In fact, the random *tendency certainty* is frequently higher than 0.6, meaning that the random method cannot prevent a user's actual characteristic behavioral tendencies from being revealed in many cases. This result demonstrates the quantitative side of the problem with the random method as illustrated in Fig.4. In contrast, the heuristic *tendency certainty* is lower than 0.2 for more than 95% of the iterations, indicating that the heuristic method can select dummy queries that are effective in hiding a user's characteristic behavioral tendencies.

Next, we evaluated the properties of dummy queries created by the heuristic method in terms of the probability distribution constructed from collected behaviors. Fig.7 shows the probability distribution of categories constructed from

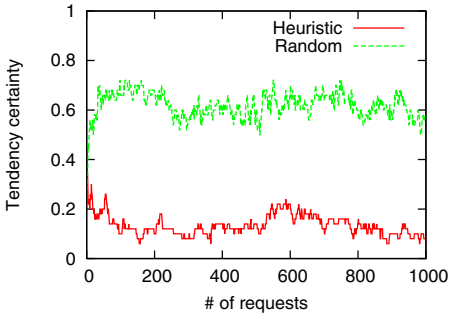


Fig. 6. Comparison of tendency certainty

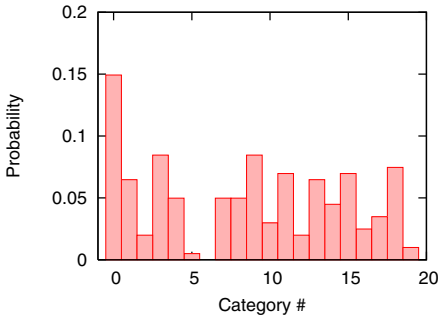


Fig. 7. Probability for actual behaviors

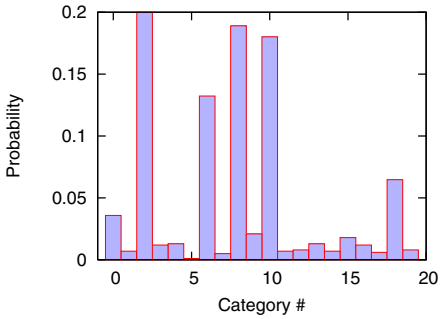


Fig. 8. Probability for dummy behaviors

Table 2. The number of queries created by the proposed and random method

ρ	0.0	0.2	0.4	0.6	0.8
Proposed	9.29	3.05	2.05	2.00	2.00
Random	17.1	10.1	4.70	2.56	2.03

actual behaviors, while Fig.8 shows the probability distribution of categories constructed from behaviors collected by the target service at the last iteration. Fig.7 shows that the highest-probability categories for the *actual tendencies*, obtained by the function $IS(Q_t^a, r)$, are $0, 3, 9, 11, and 19$. In contrast, Fig.8 shows that the highest-probability categories for the *service-side tendencies* are $2, 6, 8, 10, and 19$, thus confirming that the actual behavioral tendencies could be hidden by the heuristic method. Additionally, the categories included among the *service-side interests* have high probability, while the other categories have low probability. In Fig.8, the probability of categories contained in *service-side tendency* is 0.157 on average and the probability of the other categories is 0.0142 on average. It means that the heuristic dummy selection method selects dummy behaviors to emphasize the categories contained in *service-side tendency*, thus proposed method can meet the requirements.

6.3 Evaluation of Proposed Framework

Here, we evaluated the number of queries required to satisfy *ρ-tendency certainty*. We compared the proposed framework in the cases of the heuristic dummy selection method ("proposed") and the random dummy selection method ("random"). Because of the possibility that neither method could satisfy *ρ-tendency certainty*, we set the upper limit on the number of queries as 20. Thus, number of queries ranged from 2 to 20. We also use the experimental parameters listed in Table1 for this evaluation. Table2 lists the mean numbers of queries for each value of ρ. In comparison with the random method, the proposed method can reduce the number of queries. In particular, this trend is clear when the value of ρ is low. The proposed method can thus preserve privacy from collected

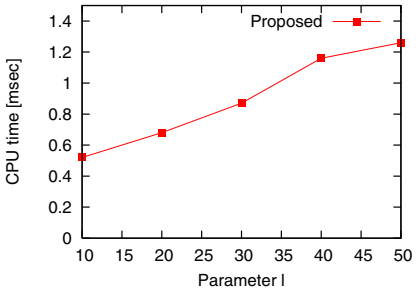


Fig. 9. Cpu time of proposed method

Table 3. The number of queries created by proposed method

l	10	20	30	40	50
Number of queries	2.52	3.05	3.25	3.56	3.09

behaviors without selecting unnecessary queries for cases when the value of ρ is at least 0.4. Consequently, our method can effectively reduce the network traffic generated by the query obfuscation approach.

Next, we evaluate the cpu time for selecting a dummy behavior by propsed method with changing a parameter l, that is number of queries that are selected randomly in the heuristic dummy selection method. Fig.9 shows a graph of the mean cpu time during selecting one dummy behavior at each value of the parameter l. Also we show the number of queries for satisfying ρ-*tendency certainty* at each value of the parameter l in Table3. While the cpu time linearly increase in Fig.9, the number of queries are almost constant in Table3. It means that when the dummy selection method selects optimal dummy query by calculating $eval(b)$ for all behavior $b \in B$ in proposed framework, it cannot reduce number of queries in spite of needing much cpu time. Thus, proposed method can effectively obfuscate user's actual behaviors and characteristic behavioral tendencies.

7 Conclusion

In this paper, we have tackled the privacy problem of using behaviors accumulated in a user's terminal by personalization services. We have focused on revealing characteristic behavioral tendencies from among behaviors collected by service providers. We proposed a model for preserving privacy, called ρ-*tendency certainty*, which considers breaches of privacy from collected information. We also proposed a behavioral tendency obfuscation framework that satisfies ρ-*tendency certainty* by sending dummy queries to service providers.

The proposed framework can effectively reduce the probability of revealing a user's actual tendencies. Additionally, this framework can reduce the network

traffic generated by sending dummy queries because it can reduce the number of dummy queries while still satisfying the model. Furthermore, this framework can quickly create dummy queries because it adopt the heuristic approach.

Thus, our framework can effectively protect pricacy in user's behaviors inputted to services. Therefore, we contribute developing the service infrastructure that user can receive more valuable personalized contents without concerning about the load of the terminal and services.

Our framework, however, is not appropriate for statistical analysis, because users' behavioral tendencies are hidden. It is limitation of our framework and we will try to resolve this problem. We have not verified the resistivity against inference attacks against our model. In a future study, we will discuss characteristics of our model further. Moreover, in our experiments, we only evaluated the proposed framework with an artificial dataset. To evaluate the practical utility of this framework, we will apply it to real user behaviors in a future work.

References

1. Sullivan, G.R.: The User of Added Error to Avoid Disclosure in Microdata Releases, PhD thesis, Iowa State University (1989)
2. Pang, H.: Obfuscating the Topical Intention in Enterprise Text Search. In: 28th International Conference on Data Engineering, pp. 1168–1179 (2012)
3. Chor, B.: Private Information Retrieval. In: 36th Annual IEEE Symposium on Foundations of Computer Science, pp. 41–50 (1995)
4. Kushilevitz, E.: Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval. In: 38th Annual Symposium on Foundations of Computer Science, pp. 364–373 (1997)
5. Kido, H., Yanagisawa, Y., Satoh, T.: An Anonymous Communication Technique using Dummies for Location-based Services. In: International Conference on Pervasive Services (ICPS 2005), pp. 88–97 (2005)
6. Lu, H., Jensen, C.S., Yiu, M.L.: PAD: Privacy-Area Aware, Dummy-Based Location Privacy in Mobile Services. In: Proc. MobiDE, pp. 16–23 (2008)
7. Song, D.X.: Practical Techniques for Searches on Encripted Data. In: IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
8. Lindel, Y.: Privacy Preserving Data Mining. Journal of Criptology 15, 177–206 (2002)
9. Adar, E.: User 4xxxxx9: Anonymizing Query Logs. In: Query Log Analysis Workshop, WWW (2007)
10. Chaum, D.L.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24, 84–90 (1981)
11. Machanavajjhala, A.: l-Diversity: Privacy Beyond k-Anonymity. In: 22th International Conference Data Engineering, pp. 24–35 (2006)
12. Li, M.: t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In: 23th International Conference Data Engineering, pp. 106–115 (2007)
13. Xiao, X.: m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets. In: International Conference on Management of Data, pp. 689–700 (2007)

A Framework for Data Processing at the Edges of Networks

Ichiro Satoh

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
ichiro@nii.ac.jp

Abstract. This paper proposes a distributed processing framework inspired by MapReduce processing. It is unique to other distributed processing approaches to large-scale data, i.e., so-called big data, because it can locally process data maintained in distributed nodes, including sensor or database nodes with non-powerful computing capabilities connected through low-bandwidth networks. It introduces mobile agent technology so that it distributes data processing tasks to distributed nodes as a map step and aggregates their results by returning them to specified servers as a reduce step. The paper describes the architecture of the framework, its basic performance, and its applications.

1 Introduction

Data processing has been one of the most typical applications of distributed systems. There have been many attempts to achieve such processing in grid computing and cloud computing. Most existing super computers have been constructed as distributed systems. This is because the amount of data required to be processed is large beyond the capabilities of individual computers. However, it is not easy for non-professional users to process data over distributed systems, since these systems have several problems and constraints, which we are not confronted with in individual computers, e.g., failures in computers and networks, and communication latencies.

Several researchers have explored several computing models for data processing over distributed systems, where the models can mask such problems and constraints as much as possible. MapReduce is one of the most typical and modern computing models for processing large data sets in distributed systems. It was originally studied by Google [3] and inspired by the *map* and *reduce* functions commonly used in parallel list processing (LISP) or functional programming paradigms. *Hadoop*, is one of the most popular implementations of MapReduce and was developed and named by Yahoo!. Stream processing has recently become a popular approach to event processing with continuous queries. It is useful for processing data generated from sensors in a real time. The original MapReduce, its clones, and stream processing assume data are transferred from the edges of the network to high performance servers or cloud computing infrastructures before they start processing. They are unable to cope cost-effectively, if at all, with new dynamic data sources, because data transmission from the nodes at the edges to the database servers at the center is costly and this traffic causes congestion in networks.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 304–318, 2013.

To solve this problem, we need to process data close to the sources of data sensors and embedded computers as much as possible. This paper proposes MapReduce-based framework processing at the edges of networks, e.g., the Internet of Things (IoTs) or machine-to-machine (M2M) environments. The goal of our MapReduce implementation was to locally execute MapReduce processing at network edges, e.g., at sensor nodes and embedded computers. The basic idea behind our implementation was to deploy data processing at the nodes that had the target data and aggregate the results, rather than to transmit data to servers at the center. It introduces mobile agent technology [8], where mobile agents are autonomous programs that can travel from computer to computer in a network, at various times and to places of their own choosing. The state of a running program is saved by being transmitted to the destination. The program is resumed at the destination and continues its processing with the saved state.

Our implementation of MapReduce defines the management system and data processing tasks as mobile agents, and *map* and *reduce* processing in MapReduce are provided by migrating workers, which are implemented as mobile agents, with the results of their processing. It is constructed based on our original mobile agent platform, which is designed for data processing, in particular MapReduce processing.

One of the most important advantages of MapReduce processing is to conceal the results of difficulties from distributed systems. The requirements for our programming model for data processing are different from other existing MapReduce implementations, because our framework must support infrastructures in the real world rather than those of data centers. In fact, the former are varied and unstable in comparison with the latter. Nevertheless, we need a framework that enables application-specific developers to easily define data processing at the edges.

This paper is organized as follows. Section 2 is a survey of related work and Section 3 explains the basic ideas behind our mobile agent-based MapReduce processing framework. Section 4 describes the architecture and processing in the framework. Section 5 presents an implementation of the middleware, Section 6 describes the basic performance of the current implementation, and Section 7 illustrates its applications. Section 8 discuss the framework and Section 9 is a summary and presents some issues for the future.

2 Related Work

The tremendous opportunities to gain new and exciting values from big data are compelling for most organizations, but the challenge of managing and transforming them into insights requires new approaches, such as MapReduce processing. It originally supported *map* and *reduce* processes [3]. The first is invoked by dividing large scale data into smaller sub-problems and assigning them to worker nodes. Each worker node processes the smaller sub-problems. The second involves collecting the answers to all the sub-problems and aggregates them as answers to the original problem it was trying to solve. There have been many attempts to improve Hadoop, which is an implementation of MapReduce by Yahoo! in academic or commercial projects. However, there have been few attempts to implement MapReduce itself except for Hadoop. For example, the Phoenix system [10] and MATE systems [6] supported multiple core processors with

shared memory. Also, several researchers have focused on iteratively executing MapReduce efficiently, e.g., Twister [4], Haloop [2], MapReduce with access patterns (MRAP) [9], and SSS [7]. These implementations, except for SSS, assume data in progress are to be stored at temporal files rather than at key-value stores in data nodes and SSS executes data stored in a key-value store shared with task nodes and then aggregates its results in the key-value store. They assume data to be stored are in high-performance servers for MapReduce processing, instead of at the edges.

Google's MapReduce, Hadoop, and other existing MapReduce implementations have assumed their own distributed file systems, e.g., the Google file system (GFS) and Hadoop file system (HDFS), or shared memory between processors. For example, Hadoop needs to move target data from external storage systems to HDFS via networks before processing these. Our MapReduce system does not move data between nodes. Instead, it deploys program codes for defining processing tasks to nodes that have data by using the migration of agents corresponding to the tasks and it executes the codes with their current local data. Hadoop and its extensions are unsuitable in sensor networks and embedded computers, because its file system, HDFS, tends to become a serious bottleneck in the operation of Hadoop and it often requires wide band networks, which may not be available in sensor nodes or embedded computers. In the literature on sensor networks, The Internet of Things (IoTs), and machine-to-machine (M2M) communications, several academic or commercial projects have attempted to support data at the edges, e.g., at sensor nodes and embedded computers. For example, Cisco's *Flog Computing* [1] and EMC's IoTs intend to integrate cloud computing over the Internet and peripheral computers. However, most of them do not support the aggregation of data generated and processed at the edge.

3 Background

We are confronted with the volume, variety, and velocity of complex, unstructured, and unstructured data, where most of the data are generated at network edges, e.g., at sensors and embedded devices. The analysis of such data provides opportunities to gain richer, deeper, and more accurate insights into our lives and business.

3.1 Requirements

Before explaining our system, let us discuss the four main requirements of systems.

- Nodes at network edges, e.g., at sensor nodes and embedded computers may have non-powerful processors with small amounts of memory, rather than data centers and high performance server clusters.
- Networks may be low-band, often disconnected, and dynamic, e.g., wireless sensor networks. Therefore, our implementation should be available in such networks.
- There may be no database or file systems in the target systems. Instead the data that need to be processed are generated or locally maintained in the local storage of nodes.
- Every node may be able to support management and/or data processing tasks, but may not initially have any codes for its tasks.

Most existing implementations of MapReduce, including extensions of Hadoop, have attempted to improve iterative processing of the same data. However, our framework is not aimed at such iterative processing. This is because most data at sensor nodes or embedded computers are processed only once or a few times. Suppose logs located at network equipment are analyzed. Only updated log data are collected and analyzed every hour or day instead of the data that have already been analyzed.

3.2 Design Principles

As previously mentioned, our system supports data processing at network edges. The system has the three main advantages:

- *Data processing at edges:* More data are generated at the edges of networks, e.g., at sensors and devices, than servers, including data centers and the cloud computing infrastructure. The transmission of such data from nodes at the edges to server nodes seriously affects performance in analyzing data and results in congestion in networks.
- *Task deployment rather than data deployment:* To solve this problem, data processing tasks are defined as mobile agents and they are dynamically duplicated and deployed at the nodes that have the target data. Mobile agents can also directly access data from sensors and low-level file systems at their destination nodes.
- *Parallel and asynchronous execution:* Data processing should be executed in parallel on nodes at the edges. Our approach assumes data at nodes are independent of one another and they can be processed without having to exchange data between nodes. Finally, like MapReduce, agents running on nodes carry their results to specified nodes after their processing is done to aggregate the results.

The framework introduces mobile agent technology as just an implementation of MapReduce processing that can satisfy the requirements and design principles discussed above. However, developers and operators for data processing do not need to support mobile agents explicitly, because mobile agents are used as entities for carrying and executing data processing tasks and results. In fact, it allows developers to define their MapReduce processing from three parts of map, reduce, and data processing, as Java classes, which can satisfy specified interfaces. The map and reduce classes have similar methods in the `Mapper` and `Reducer` classes in Hadoop. The data processing parts are responsible for data processing at the edges. They consist of three methods corresponding to three functions of reading data locally from nodes at the edge, processing the data, and storing the results in a key-value store format.

There have been several approaches to deploying programs for data processing on nodes at edges, e.g., code migration techniques and code caching. We therefore need to explain the reason why we used mobile agent technology as just an implementation of MapReduce processing.[1] Mobile agent and simpler approaches, e.g., code migration, almost have no differences in the cost of deploying and executing task programs. Nevertheless, the former has several advantages, which the latter does not.

[1] Unlike other software agents, mobile agents have been used as just a technique of distributed systems. We did not intend mobile agents to support any intelligent behaviors.

- Mobile agents can dynamically select one or more destinations and migrate to these according to the results of computations, unlike other approaches, including code migration and technique. This is useful to deploy programs for data processing and gather results from nodes at edges, where connections between such nodes tend to be unstable and change dynamically.
- They can be managed in a decentralized manner. This is important to process data stored on numerous nodes for reasons of scalability and reliability.
- They can automatically store and resume data in heap memory before and after deploying task programs so that developers can concentrate on their application-specific data processing without knowledge of distributed systems like Hadoop.

Some readers may think that task programs can be cached or pushed at the edges of networks. However, since we did not intend to be iteratively executed processing MapReduce, tasks cannot be expected in advance.

4 Mobile Agent-Based MapReduce

This section outlines our mobile agent-based MapReduce processing system and compares between our system with Hadoop, which one is of the most typical implementations of MapReduce. The architecture of our MapReduce system is different from existing implementations of MapReduce, including Hadoop.

4.1 Architecture

The original MapReduce consists of one *master* node and one or more *worker* nodes and Hadoop consists of a *job tracker*, *task tracker*, *name*, and *data* nodes, where the first and third corresponds to the master node, and the second and fourth to the data nodes in the original MapReduce. Our MapReduce system has a slightly different architecture from Google's MapReduce, which is quite different from Hadoop's. The system itself is a collection of three kinds of mobile agents, called *Mapper*, *Worker*, and *Reducer*, which should be deployed at appropriate nodes, called *mapper nodes*, *data nodes*, and *reducer nodes* respectively. Not only *Worker* agents but also *Mapper* and *Reducer* agents can dynamically be deployed at nodes according to the location of target data and available resources to process them.

4.2 MapReduce Processing

Our system supports MapReduce processing with mobile agents. Figure 1 outlines the basic mechanism for processing, with involves five steps.

Step 1 A *Mapper* agent corresponds to the master node of the original MapReduce. The agent makes copies of *Worker* agents.
Step 2 Each of the *Worker* agents migrates to one or more data nodes, which locally have the target data.
Step 3 Each of the *Worker* agents executes its processing at its current data node.

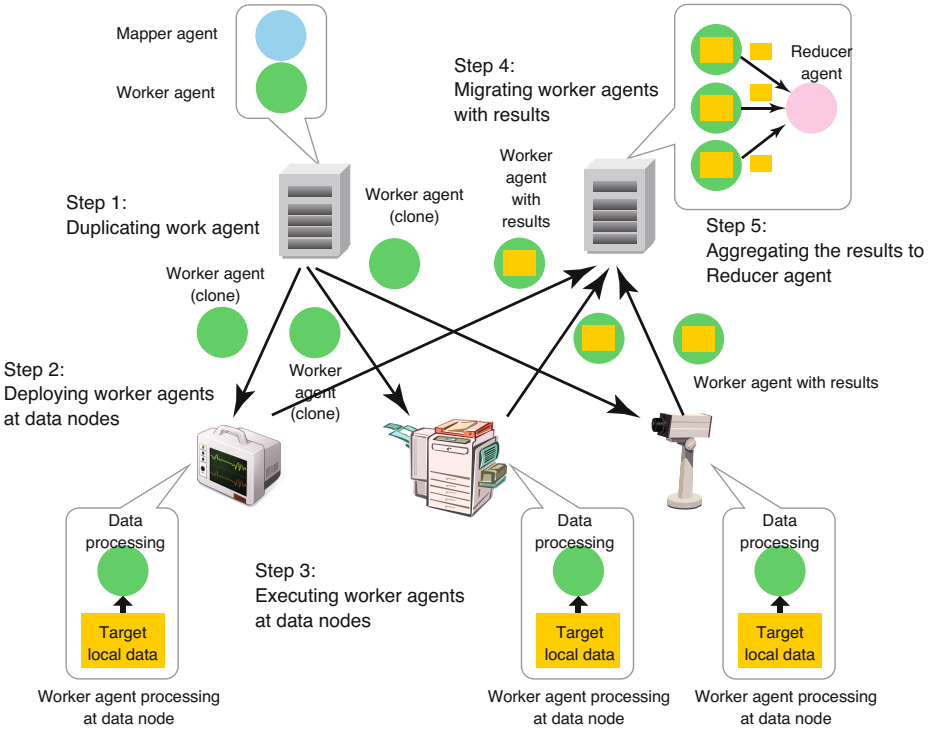


Fig. 1. Mobile agent-based MapReduce processing

Step 4 After they execute their processing, *Worker* agents migrate to the computer that the *Reducer* agent is running on with their results
Step 5 Each *Worker* agent sends its results to the *Reducer* agent.

Note that the number of results is by far smaller than the amount of target data. Each *Worker* agent assumes it is to be executed independently of the others. *Mapper* and *Reducer* agents can be running on the same node.

4.3 Enhancement of MapReduce Processing

Mobile agent technology can extend MapReduce processing with two mechanisms:

- Mobile agents can migrate between computers through their own itineraries. Our *Worker* agents can decide their next destinations according to their results of processing. For example, if *Worker* agents cannot find data that they want at their current data nodes, they can migrate to other nodes until they achieve their goals.
- Since *Mapper* and *Reducer* agents in our approach are implemented as mobile agents like *Worker* agents, this means that our approach allows *Mapper* and *Reducer* agents to migrate between computers. Also, we do not need to distinguish between *Mapper*, *Reducer*, and *Worker* agents. Therefore, *Worker* agents can become *Mapper* or *Reducer* agents. Therefore, our approach enables each worker task to work as MapReduce processing.

5 Design and Implementation

This section describes our mobile agent-based MapReduce system. It consists of two layers, i.e., mobile agents and runtime systems. The former consists of agents corresponding to job tracker and *map* and *reduce* processing and the latter corresponds to task and data nodes. It was implemented with Java language and operated on the latter with Java virtual machine (JVM). The current implementation was built on our original mobile agent platform, because existing mobile agent platforms are not optimized for data processing and need the developers for data processing to have knowledge about mobile agent processing.

5.1 Runtime System

Each runtime system runs on a computer and is responsible for executing *Mapper*, *Worker*, and *Reducer* agents at the computer and migrating agents to other computers through networks (Fig. 2). The system itself is designed independently of any application-specific data processing. Instead, agents running on it support MapReduce processing. Each runtime system is light so that it can be executed on embedded computers, including JVMs for embedded computers.

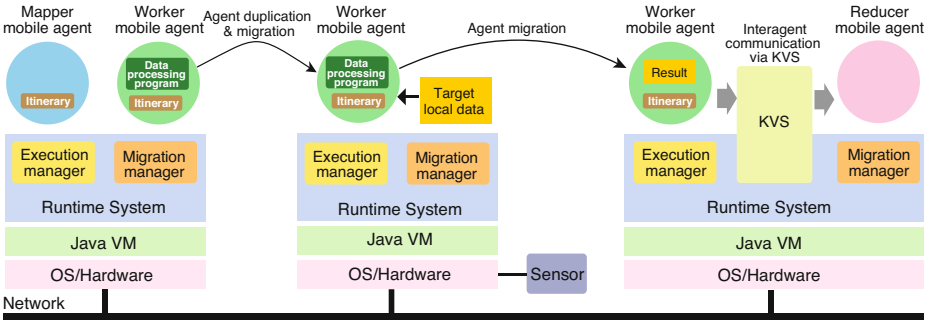


Fig. 2. Runtime systems for mobile agent-based MapReduce processing

Task Duplication. Our approach is used to make one or more copies of *Worker* agents before agents are deployed at data nodes. The runtime system can store the states of each agent in heap space in addition to the codes of the agent into a bit-stream formed in Java's JAR file format, which can support digital signatures for authentication. The current system basically uses the Java object serialization package for marshaling agents. The package does not support the capturing of stack frames of threads. Instead, when an agent is duplicated, the runtime system issues events to it to invoke their specified methods, which should be executed before te agent is duplicated, and it then suspends their active threads.

Task Migration. Each runtime system also establishes at most one TCP connection with each of its neighboring systems in a peer-to-peer manner without any centralized management server and it exchanges control messages and agents through the connection. When an agent is transferred over a network, the runtime system transfers the agent in bit-stream like task duplication and transmits the bit-stream to the destination data nodes through TCP connections from the source node to the nodes. After they arrive at the nodes, they are resumed and activated from the marshalled agents and then their specified methods are invoked to acquire resources and they continue processing.

Task Execution. Each agent can have one or more activities, which are implemented by using the Java thread library. Furthermore, the runtime system maintains the life-cycles of agents. When the life-cycle state of an agent is changed, the runtime system issues certain events to the agent. The system can impose specified time constraints on all method invocations between agents to avoid being blocked forever. Each agent is provided with its own Java class load, so that its namespace is independent of other agents in each runtime system. The identifier of each agent is generated from information consisting of its runtime system's host address and port number, so that each agent has a unique identifier in the whole distributed system. Therefore, even when two agents are defined from different classes whose names are the same, the runtime system disallows agents from loading other agents's classes. To prevent agents from accessing the underlying system and other agents, the runtime system can control all agents under the protection of Java's security manager.

5.2 Programming Model

Our framework supports data processing on nodes, e.g., sensor nodes and embedded computers, which may be connected through non-wideband and unstable networks, whereas existing MapReduce implementations are aimed at data processing on high-performance servers connected through wideband networks. Therefore, we cannot directly inherit a programming model from existing MapReduce processing.

In comparison with other MapReduce processing, including Hadoop, our framework explicitly divides the *map* operations into two parts in addition to a part corresponding to the *reduce* operation in MapReduce.

- *Duplication and deployment of tasks at data nodes* Developers specify a set of the addresses of the target data nodes that their data processing has executed or the the network domains that contain the nodes. If they still want to define more complicated MapReduce processing, our framework is open to extend the *Mapper* and *Reducer* agents.
- *Application-specific data processing* They define the three functions of reading data locally from nodes at edges, data processing the data, and storing their results in a key-value store format. These functions can be isolated so that developers can define only one or two of the functions according to the requirements of their data processing.
- *Reducing data processing results* They define how to add up the answers of data processing stored in a key-value store.

Although the first is constructed in *Mapper* and *Worker* agents, the second in only *Worker* agents, and the third in *Worker* and *Reducer* agents, developers focus on these three parts independently of their mobile agent-level implementations. Each *Mapper* agent provides an itinerary to its *Worker* agent, where each itinerary is defined as a combination of the basic itinerary patterns shown in Fig. 3.

- The first is to instruct the *Worker* agent to duplicate itself and then and instruct the duplicated *Worker* agents to migrate to and execute their data processing on the specified data nodes.
- The second is to instruct the *Worker* agent to migrate to and execute its data processing on one or more specified data nodes.
- The third is to instruct the *Worker* agent to go back and forth between each of the data nodes and the source node (or the reducer node) to execute its data processing on the nodes.
- The fourth is to instruct one or more *Worker* agents to migrate to the reducer node and then pass their results to the *Reducer* agent at the reducer node.
- The fifth is to instruct the *Reducer* agent to migrate to one or more data nodes to receive the results of the *Worker* agents on the data nodes and then go back to the reducer node.
- The sixth is to instruct the *Reducer* agent to go back and forth between each of the data nodes and the reducer node to receive the results of the *Worker* agents on the data nodes.

Our system enables us to easily define application-specific *Mapper*, *Reducer*, and *Worker* agents as subclasses of three template classes that correspond to *Mapper*, *Reducer*, and *Worker* respectively, with several libraries for KVS. When an *Mapper* agent gives one or more *Worker* agents no information, we can directly define the agent from the template class for *Mapper*. It can create specified application-specific *Worker* agents according to one or more specified data and deploy them at the nodes. When *Reducer* agents support basic calculations, e.g., adding up, averaging, and discovering maximum or minimum values received from one or more *Worker* agents through KVSs according to the keys, we can directly define them as our built-in classes.

5.3 Key-Value-Store

MapReduce processing should be executed on each of the data nodes independently as much as possible to reduce the amount of data transmitted through networks. However, data may not be divided into independent pieces. Hadoop enables data nodes to exchange data with one another via HDFS to solve this problem, because HDFS is a distributed file system shared by all data nodes in a Hadoop cluster, like the GFS.

Unlike other existing MapReduce implementations, including Hadoop, our system does not have any file system, because nodes in sensor networks and ambient computing systems may lack enriched storage devices. Instead, it provides tree-structured key value stores (KVSs), where each KVS maps an arbitrary string value and arbitrary byte array data and is maintained inside its agent, and provides directory servers to KVSs in agents. The root KVS merges the KVS of agents into itself to support *reduce* processing.

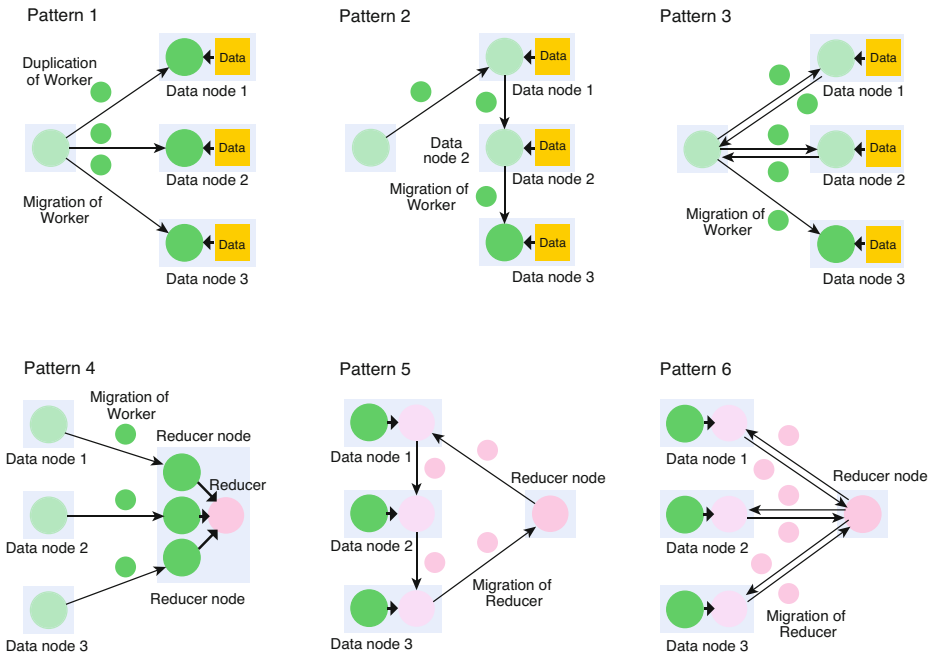


Fig. 3. Basic itinerary patterns

Each KVS in each data processing agent is implemented in the current implementation as a hashtable whose keys, given as pairs of arbitrary string values, and values are byte array data and it is carried with its agent between nodes.

Whenever an agent corresponding to a *job tracker* in Hadoop starts MapReduce processing, it creates one or more processing agents and a root KVS and assigns the references of the KVSs of the newly created processing agents to the root KVS. Each directory server can run in the external system from agent runtime systems and agents or inside an agent corresponding to a *job tracker*. It tracks the current locations of mobile agents so that it can enable an agent to access KVSs maintained in other agents, which may move to other destinations, with the identifiers of the moving agents. To support *reduce* processing, the root KVS merge KVS of agents into itself. Each KVS in each data processing agent is implemented as a hashtable in the current implementation, whose keys, given as pairs of arbitrary string values and values, are byte array data and it is carried with its agent between nodes. However, a KVS in a *job tracker* agent is also implemented as a hashtable whose keys are given as pairs of identifiers of the agents that its agent creates, and the values are references to them.

5.4 Job Scheduling

MapReduce can be treated as batch processing over distributed systems. If there is more than one *Mapper* agent in our system, they can be running independently. Therefore, we introduce a *schedule* agent between running *Mapper* agents, if we need to manage

the whole system. The agent is responsible for controlling *Mapper* agents and monitoring *Reducer* agents. When it also detects the completeness of *Reducer* agents, it can explicitly send a *start* message to one or more *Mapper* agents to instruct them to start processing.

5.5 Fault-Tolerance

The job manager in Hadoop is responsible for supporting fault tolerances against crash failures in data nodes. The manager detects failures in data nodes, because each *task tracker* running on a data node sends heartbeat messages to to the *job tracker* every few minutes to inform of its status. Since data are shared by worker nodes, the *job tracker* pushes work out to available *task tracker* nodes in the cluster, striving to keep the work as close to the data as possible.

However, our system assumes that data are maintained in one data node so that it has a different policy for fault tolerances. If a data node is stopped or disconnected, it needs to exclude such a node. Our system introduces a mobile agent-based *job tracker* manager, called a *system manager* agent, which has a Java Management Extension (JMX) interface to monitor data nodes and it periodically sends messages to data nodes. When they receive a message, data nodes returns their status to the *system manager* agent.

- If a data node has crashed, the *system manager* agent informs *Mapper agents* to leave out the crashed node from the list of the target data nodes, before the *Mapper* agent dispatches *Worker* agents.
- If a data node has crashed, the *system manager* agent informs the *Reducer* agent to leave out agents returned from nodes from the agent's waiting list, after *Worker* agents are deployed at the target node. Even when the crashed node can be restarted or it continues to work, the *Reducer* agent does not wait for any agents from the node.

The *system manager* agent can explicitly make clones of these agents at other nodes because they are still mobile agents to mask failures in in the nodes that runs *Mapper* and *Reducer* agents.[2] The current implementation has no fault-tolerant mechanisms for failures while *Worker* agents are deployed and running, because our MapReduce processing is not heavy. We should restart the processing again.

5.6 Security

The current implementation is a prototype system to dynamically deploy the components presented in this paper. Nevertheless, it has several security mechanisms. For example, it can encrypt components before migrating them over the network and it can then decrypt them after they arrive at their destinations. Moreover, since each component is simply a programmable entity, it can explicitly encrypt its individual fields and migrate itself with these and its own cryptographic procedure. The JVM could explicitly restrict components so that they could only access specified resources to protect

[2] The current implementation does not support consistency between original agents and their clones.

computers from malicious components. Although the current implementation cannot protect components from malicious computers, the runtime system supports authentication mechanisms to migrate components so that all runtime systems can only send components to, and only receive them from, trusted runtime systems.

6 Performance Evaluation

A prototype implementation of this framework was constructed with Sun's Java Developer Kit version 1.5 or later versions. The implementation provided graphical user interfaces to operate the mobile agents. Although the current implementation was not constructed for performance, we evaluated that of several basic operations in a distributed system where eight computers (2 GHz Intel Core Duo 2 with MacOS X 10.7 and J2SE version 6) were connected through a Giga Ethernet.

- The cost of agent duplication was measured as plotted at the left of Fig. 4, where The agent was simple and consisted of basic callback methods. The cost included that of invocating two callback methods.
- The cost of migrating the same agent between two computers was measured as plotted at the right of Fig. 4. The cost of agent migration included that of opening TCP transmission, marshaling the agents, migrating the agents from their source computers to their destination computers, unmarshaling the agents, and verifying security.

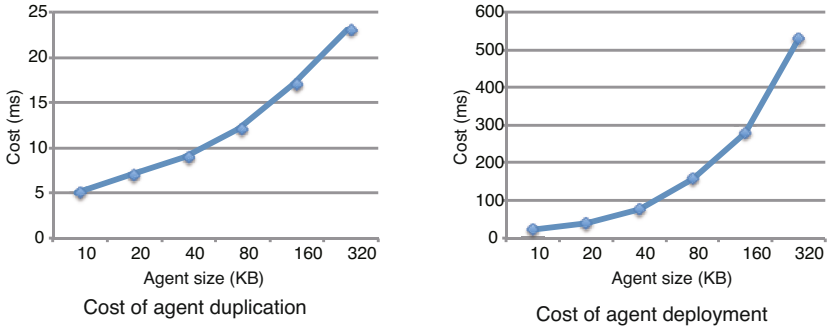


Fig. 4. Costs of agent duplication and deployment in MapReduce processing

7 Applications

The first application of our system was word counting in texts.[3] Figure 5 shows the basic structure for our word counting application by using mobile agent-based MapReduce with screenshots of word counting. The system consisted of four computers, i.e., *Mapper*, two *Data*, and *Reducer* nodes. The first executed an instance defined as a *Mapper* agent, which created two *Worker* agents for word counting deploying them at the two data nodes. The second was two data nodes, where the first had a text file of

[3] Word counting is one of the most typical examples of Hadoop.

Shakespeare's play, Romeo and Juliet (size of about 100 1KB) and the second had a text file of his play Macbeth (size of about 138 KB). The data nodes executed *Worker* agents for word counting, where each of the agents read the text file maintained in its current node, i.e., Romeo and Juliet or Macbeth, and counted how often words occurred in the file. The third executed a *Reducer* agent, which added up the values of each of the words received from the two *Worker* agents for word counting obtained from the *Data* nodes. Figure 5 shows the basic steps of counting how often words occurred in the two files.

The cost of word counting for the two files was 520 ms in our system, where the cost of doing this in Hadoop was 4.8 s, because Hadoop was originally designed for coarse batch processing of large amount of data and it needed to transmit the files to its HDFS in addition to processing.

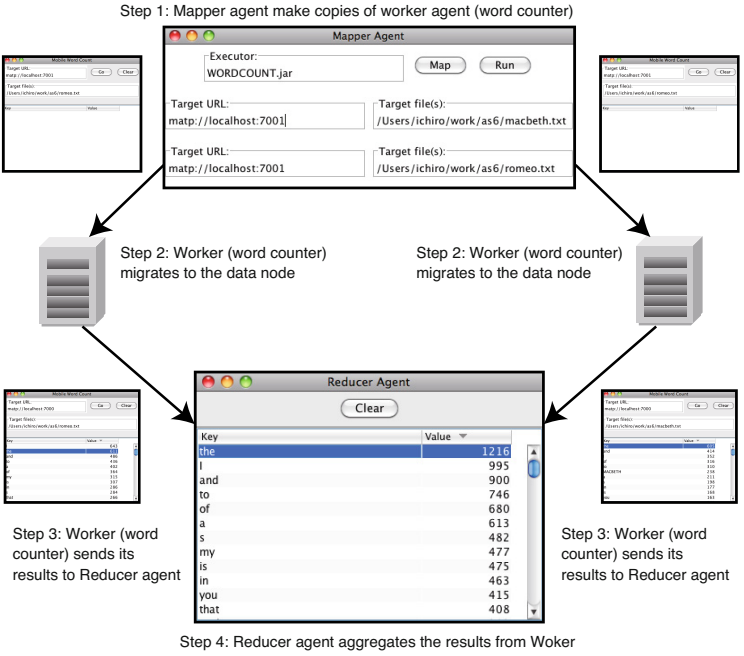


Fig. 5. Mobile agent-based MapReduce processing

The second application of our system was a detection of abnormalities from data measured by sensors. It detected anomalous data, which were beyond the range of specified maximum and minimum values. This evaluation assumed each data node would have 0.01 % of abnormal data in its stream data generated from its sensor every 0.1 second and each data entry would be 16 bytes. We detected abnormal values from a data volume corresponding to a data stream for one year at each of eight data nodes. The whole amounts of data was about 5.04 GB and the amount of abnormal data was 504 KB in each node. When we used Hadoop, we needed to copy about 40 GB of the data, i.e., multiply 5.04 GB by eight, from data nodes to HDFS. The cost of the entire processing was 232 s.

We compared our system with Hadoop-based system. Our experimental system for Hadoop consists of Intel Core 2 Duo processors (2 GHz) with 4GB of memory, and 7200 rpm hard disk drives and it could copy data from data nodes to its HDFS with about 28 MB/s throughput via Giga Ethernet.[4] The total cost of data transmission from the data nodes to HDFS was evaluated as 184 s. The cost of data processing in Hadoop was about 201 s. The whole cost of Hadoop was 352 s. This application in our system was useful to thinning out unnecessary or redundant data from the large amount of data stored at the edges of networks before data were transmitted through networks. In fact, the application could reduce from 40 GB of data to 4 MB.

8 Discussion

Our framework was used to deploy programs for data processing at nodes, e.g., sensor nodes and embedded computers, that stored target data, where existing MapReduce implementations deployed the data, GFS and HDFS that were shared by servers that executed their processing. Our framework had several advantages. For example, there was no cost to deploy data, which tended to be huge, to the file system. In fact, it outperformed Hadoop. It could process data at original nodes so that it could process data that could not be removed for the reasons of security and access limitations. Its architecture was simple because its whole processing was constructed from a combination of *Mapper*, *Worker*, and *Reducer* agents rather than an underlying centralized management system. As a result, it could support light-weight data processing. However, it had several disadvantages. It assumed that data processing on each data node was isolated from other nodes, because it lacked any mechanisms to exchange data between data nodes. However, data processing at nodes can be executed in isolation in our potential applications, e.g., data collection from sensor nodes. It was not suitable for iteratively executing MapReduce processing for the same data.

9 Conclusion

We presented a distributed processing framework based on MapReduce processing. It was designed for analyzing data at the edges of networks and was constructed based on mobile agents. It could distribute programs for data processing to nodes at the edges as a *map* operation, execute the processing to gather and analyze data maintained at the nodes, and then collect their results from the nodes as a *reduce* operation.[5] The approach could provide application-specific processing of data at the edges of networks, e.g., at sensor nodes and embedded computers, more cost-effectively than existing MapReduce processing and its clones, including Hadoop. As previously mentioned, our framework is useful for thinning out unnecessary or redundant data from the large amounts of data stored at the edges of networks, e.g., at sensor nodes and embedded computers

[4] Throughput was bound by performance of individual data nodes.

[5] The runtime system and the second application can be downloaded as open source software from `http://research.nii.ac.jp/~ichiro/agent/agentspace_v3/index.html`.

connected through low-bandwidth networks, before data are transmitted to processing systems at the center, e.g., MapReduce-based systems. It enables developers to define application-specific data processing at the edges without any knowledge of mobile agents as well as distributed systems.

In concluding, we would like to identify further issues that need to be resolved. Our system should be more compatible with Hadoop's classes and interfaces, e.g., `Mapper` and `Reducer`, to directly reuse existing data processing software for Hadoop. We have many issues to resolve to improve the performance of the system. For example, the current implementation tended to have bottlenecks in *Reducer* agents, because they needed to sequentially merge results from multiple *Worker* agents via our KVS database. We plan to extend the KVS database with the ability to record results in parallel. The current implementation assumes nodes are connected through wired networks, but we need scheduling mechanisms for wireless networks.

References

1. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of MCC Workshop on Mobile Cloud Computing (MCC 2012), pp. 13–16. ACM Press (2012)
2. Bu, Y., Howe, B., Balazinska, M., Ernst, M.D.: HaLoop: Efficient Iterative Data Processing on Large Clusters. Proceedings of the VLDB Endowment 3(1) (2010)
3. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation, OSDI 2004 (2004)
4. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.H., Qiu, J., Fox, G.: Twister: a runtime for iterative MapReduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC 2010). ACM (2010)
5. Grossman, R., Gu, Y.: Data mining using high performance data clouds: experimental studies using sector and sphere. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), pp. 920–927. ACM (2008)
6. Jiang, W., Ravi, V.T., Agrawal, G.: A Map-Reduce System with an Alternate API for Multi-Core Environments. In: Proceedings of 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (2010)
7. Ogawa, H., Nakada, H., Takano, R., Kudoh, T.: SSS: An Implementation of Key-Value Store Based MapReduce Framework. In: Proceeding of IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 754–761 (2010)
8. Satoh, I.: Mobile Agents. In: Handbook of Ambient Intelligence and Smart Environments, pp. 771–791. Springer (2010)
9. Sehrish, S., Mackey, G., Wang, J., Bent, J.: MRAP: A Novel MapReduce-based Framework to Support HPC Analytics Applications with Access Patterns. In: Proceedings of High Performance Distribute Computing, HPDC 2010 (2010)
10. Talbot, J., Yoo, R.M., Kozyrakis, C.: Phoenix++: modular MapReduce for shared-memory systems. In: Proceedings of 2nd International Workshop on MapReduce and its Applications (MapReduce 2011). ACM Press (2011)

STRING: Social-Transaction Routing over a Ring

Idrissa Sarr[1], Hubert Naacke[2], and Abderrahmane Ould Mohamed Moctar[1]

[1] University Cheikh Anta Diop - LID Laboratory, Dakar, Senegal
idrissa.sarr@ucad.edu.sn, abderrahmane.md.moctar@hotmail.fr
[2] UPMC Sorbonne Universités - LIP6 Laboratory, Paris, France
hubert.naacke@lip6.fr

Abstract. A key requirement for social applications is to support fluid interactions among users. Basically, social applications deal with user-oriented data: users own their data and may access or modify data owned by others (say their friends). Therefore, several users may focus simultaneously on a small piece of data (hot data) owned by the same user. Such a situation, more known as a net effect, has the drawback to generate temporal peak loads able to slow user interactions. Moreover, a social application inherently generates multi-node (or multi-partition) transactions as far as users interact between them.

Based on those observations, we propose *String*, a transaction scheduling layer that uses various strategies to order (or group) transactions based on their access classes. *String* reduces significantly the overhead cost of processing one transaction at a time while allowing to process rare multi-nodes transactions in en efficient way. The key novelties lie in (1) our distributed transaction scheduling devised on top of a ring to ease communication and (2) our ability to absorb peak loads as early as possible by splitting the transaction processing in two phases: a scheduling phase, resilient to peak load, followed by a group execution phase. We designed and simulated *String* using SimJava and we ran a series of experiments. Compared with some existing solutions, *String* shows interesting and promising results.

Keywords: Transaction processing, load-aware query routing, data consistency.

1 Introduction

Social applications are becoming more and more popular in the realm of computer science, economic, politic, and so forth because they give a primary role to the user, capturing its profile and supporting its collaborative activity. They bring to the user an enriched experience that capitalizes its her/his social enrollment, more than ever. A key requirement for social applications is to support fluid interactions among users. Social applications deal with user-oriented data: a user owns some items and can add and/or modify the attributes related to the items owned by others. Such applications are both read and write intensive since

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 319–333, 2013.

(1) the number of users is very important (tens of thousands) and (2) almost all user actions cause data read, insert and update (say a huge number of small transactions). Even though, transactions are usually short, they access to a small dataset regarding the size of the whole database. Moreover, many transactions may attempt to access the same dataset simultaneously (i.e., at the same time), which generates temporal peak loads on some hot data. Such a situation, more known as a net effect, has the drawback to slow user interactions.

Furthermore, social applications are characterized by a various kind of social interactive actions such as virtual game, chatting, tagging and so on. We say a social transaction any set of operations (insert/update) executed on some data for ensuring social interactions between users. Among the multiple requirements of social applications we retain three: response time, scalability and availability. Roughly speaking, an obvious way to achieve scalability is to process independent transactions in parallel manner. That is, data are partitioned or replicated over a set of nodes, which are loosely coupled and then leading to each of them to manage its own processing resource. Therefore, nodes can execute disjoint transactions in a complete parallel way. However, partitioning data perfectly in such a way that each transaction is executed on only one node is quite impossible. Thus, multi-nodes transactions, even if they are not frequent, must be managed efficiently to avoid compromising the positive effect of splitting and/or replicating data. Our main goal is to deal with peak load problem in social applications while considering multi-node transactions that may be received.

1.1 Motivations and Approach

To face social application requirements, we design *String*, a two steps approach, i.e. a scheduling step followed by an execution step. Our main objective is to design a scheduling solution which ensures low latency when a peak load occurs. The main idea is to absorb the peak load, as early as possible, during the scheduling step, before processing the transactions. Actually, the transactions are ordered during the scheduling step in such a way that each transaction is followed or preceded by another one. Afterwards, each transaction is executed at the nodes storing the required data. A node-to-node coordination mechanism guaranties that the execution order remains identical for all the nodes involved.

This two steps approach has been demonstrated to be efficient for write intensive workloads made of short transactions [13,8]. However, existing solutions following this approach assume that the workload is mostly composed of single node transactions (*i.e.,* they mostly access independent data). Existing solutions are not designed to face a peak load of concurrent transactions. In such situation, they would suffer from communication overhead and high latency.

The proposed approach for overcoming limits in existing solutions, is guided by the fact that grouping transactions with similar patterns of data access might save a significant amount of work. In other words, we want to stick together the concurrent transaction requests for faster subsequent group-execution. We leverage on the ability to process a group of concurrent transactions that is faster than processing one transaction at a time. Grouping transactions improves the

performance for two main reasons: (1) it allows for group commit ([6]) which is faster because only one logging information is produced for the group. (2) in a distributed context (several application nodes connected with a database), it reduces the number of messages sent to the database node. More precisely, if every application directly connects to the database, then the latter may become a bottleneck and overloaded. Rather, application requests can be gathered and grouped at a middleware stage and thereby, database connection requests are minimized.

1.2 Contributions and Paper Organization

The main contributions of this paper are: (1) a distributed transaction routing approach using a ring to reduce response time while ensuring global consistency, (2) a mechanism for grouping transactions based on three algorithms – a ring-based, a piggybacking-based and time-based algorithms and (3) a thorough experimentation through simulation to assess our approach and its feasibility. The remainder of this paper is organized as follows. In Section 2 we review some works connected to ours. In Section 3 we lay out our architecture and the communication model between the different pieces of our solution. Section 4 contains our distinct algorithms to deal with the management of the ring, the grouping process and transaction execution model. In Section 5 we present the results of our experiments while in Section 6 we conclude and present our future work.

2 Related Work

Our work settles in a domain of transaction processing for scalable data stores. Large scale solutions for managing data are facing a consistency/latency tradeoff as surveyed in [1]. Today several solutions relax consistency for better latency ([12,9] and do not provide serializable execution of concurrent transactions. Other solutions provide strong consistency but only allow transactions restricted to a single node [5,4]. Finally, the solutions such as [2,14] support transactions accessing any data but at the prohibitive cost of running a distributed transaction protocol. As far as we know, [13] recently provides a consistent execution of transaction accessing any data over a distributed data store without running a distributed transaction protocol. Rather, the authors design a distributed sequencing mechanism which serializes the transactions. We briefly review the classes of sequencing solutions depending on two dimensions: (i) the number of entry points that are handling the input workload (centralized / decentralized) and (ii) the transaction type (single node / multi nodes). For each class, we briefly present the sequencing principles, and discuss its behavior when peak load occurs.

1. Centralized workload of any transaction type. The sequencing is straightforward if all the concurrent transactions are centralized at one node, using queuing. However, this solution lacks availability at large scale because the

single point of access only provides a limited throughput. If the peak load exceeds the provided throughput, then the system does not serve the transactions in bounded time.

2. Decentralized workload of single node transactions. In this case, the workload is strictly partitioned since a transaction arrives to the node containing the right data. The sequencing is straightforward: each node maintains the local order for the transaction it receives. However, this specific case does not cover our social context because it does not handle multi node transactions. This sequencing method will still lack availability (i.e., low latency) when peak load occurs, i.e., when the transaction requests concentrates to few specific partitions.
3. Decentralized workload of multi-node transactions. This is the case for social workloads we are targeting. Although the workload is not strictly partitioned, the sequencing is possible. Assuming that the nodes are ordered (i.e. $N_i < N_j \; \forall i < j$), then, the sequencing is distributed as follows ([13]). Periodically, all the nodes transmit their incoming sequence of transactions to the other nodes (only these nodes) that are concerned by the sequence. Then, each node is able to assess the same ordering, in a decentralized way, for all the transactions that concern it. This solution is not robust to peak load for the following reason: consider a multi-node transaction that comes to any node that the transaction intend to access. When a transaction peak occurs, several multi node transactions are accessing some partitions in common, as well as some disjoint partitions. All these partitions may receive the transactions. Therefore, two concurrent transactions will not necessarily come to the same node. One may observe that the transaction peak tend to scatter among the partition nodes. However, even if the transactions requests are balanced over several nodes, the number of sequencing messages that comes to a node may grow up to n (for n+1 nodes): that produces a severe sequencing peak resulting in high latency.

Our work also relies on sequencing and scheduling to guaranty consistent processing of multi-node transactions, while better supporting high peak load.

3 Architecture and Design Model of STRING

We design our architecture using two layers: the scheduling layer and the execution layer (see Figure 1). That is, our solution is a middleware that serves as an interface with the data manipulation procedures of applications. The scheduling layer is made of a set of nodes called scheduling nodes (SN) while the execution layer contains execution nodes (XN). As one can see, transactions may be sent to any point and afterwards they are gathered based on their access classes for execution. For example, transactions T_B, T_{BC} and T_{AC} are grouped on the first SN_1 even if they were received by SN_2 and SN_k. More details of how transactions are grouped are described in Section 4.

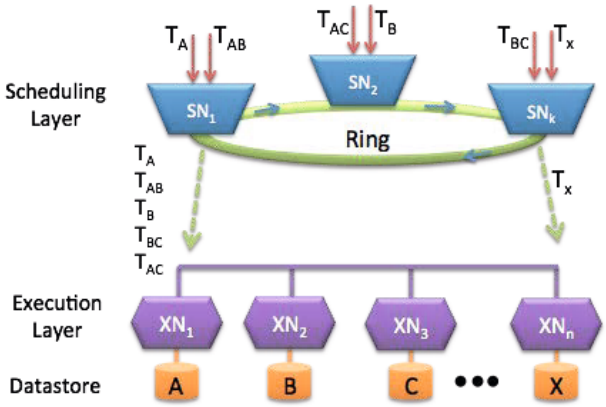


Fig. 1. The layered architecture of STRING

3.1 Scheduling Layer

The SNs are structured over a ring and receives any transaction request from applications. The scheduling layer is responsible to absorb the peak loads and serves as a front-end that isolates the underlying datastore nodes from the input peak load. Basically, the number of queries that a SN receive must remain bounded. To this end, we efficiently balance the load over all SNs regardless of the data requested by a transaction. Moreover, we assess the number of SNs according to the entire workload submitted to the system, expressed in number of clients C, and according to the the number of transaction requests that a SN is able to handle in a unit of time. The ring size is not correlated with the number of datastore nodes; rather it depends on the number of users generating the workload.

3.2 Execution Layer

An XN executes the transactions sent by a SN in a serial way. Since concurrency is already controlled at the scheduling layer, the XNs guarantee consistent execution of transactions without locking. The overall storage supports of the XN nodes forms the distributed data store. By considering a database divided into n partitions $\{p_0, p_1, ...p_{n-1}\}$ such that $p_i \cap p_j = \emptyset$, we store each p_i on at least one XN. Each XN may contain more than one partition and can be accessed by SNs. To this end, an SN uses a data access API provided by the distributed data storage layer (see Figure 1). We distinguish two storage options: (i) the storage layer is a white-box, thus a SN node may directly communicate with a well identified XN; (ii) the storage layer is a black-box, hidding the data distribution schema. In such case, we associate any partition with a key and access a partition using *get(key)* and/or *put(key, value)* primitives call supported by most of the existing datastores (e.g., Bigtable [4] and Hbase [3]).

3.3 Communication Model

SN nodes are organized into a ring as stated in Figure 2. Each SN node knows its successors and may communicate with them through a token that is a data structure. We handle two useful and distinct tokens: *processing token* and *forwarding token*.

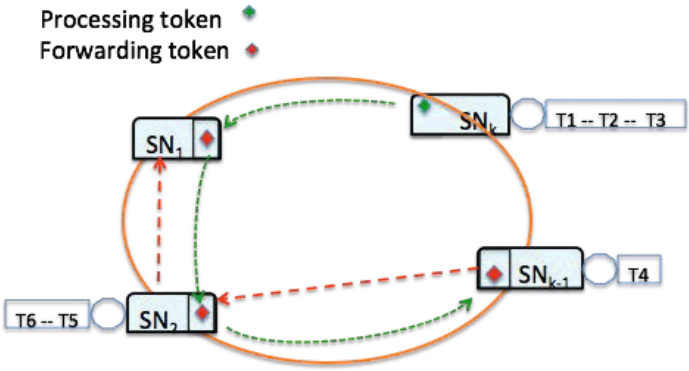


Fig. 2. Middleware layer of String

Processing Token. A processing token is used to synchronize SN nodes that desire to access the same data at a time. A token circulates in a counter-clockwise direction for giving to each SN a full access to the metadata required to route queries. Actually, a processing token τ_{p_i} is created for each partition p_i and contains metadata related to p_i. When receiving τ_{p_i}, an SN processes all transactions inquiring an access to p_i and releases the token afterwards for its immediate successor. Once a SN releases a token, it has to wait for a complete round of the token. Moreover, a pending list is maintained by each SN to queue transactions requiring an access to p_i. That is, a SN may keep a transaction into the pending list until it gets the corresponding processing token (τ_{p_i}).

Figure 2 depicts a situation with a single partition and four SNs. At the beginning, SN_k acquires the processing token (green diamond), and therefore, has to routes {T1, T2, T3} that require the same data. Once the routing process of its pending list completed, it releases the token for SN_1, which has no waiting transactions. In other words, SN_1 does not keep the token and let it go directly to SN_2, which may start doing a job for {T5, T6}.

Forwarding Token. A forwarding token moves in a clockwise direction and is used as a transport tool by a SN to send data or transactions to another SN. Each SN_i may have its own forwarding token (say f_i) or share a unique forwarding token with its peers. Whether a unique or several forwarding tokens depends on some context we describe in Section 4.2. Conversely to a processing token, a forwarding one is used only to ease a communication between two SNs for forwarding transactions. For example, in Figure 2, SN_{k-1}, may use its

forwarding token (red diamond) to send T4 directly to SN_2 or SN_k in order to accelerate the execution of T4. When and why a transaction is sent from one SN to another is detailed in Section 4.1.

4 Dealing with Transactions

In this section we describe the transaction routing scenario used by highlighting what the scheduling layer does for incoming transaction. Afterwards, we describe various mechanisms to schedule transactions for reducing the routing time as well as the execution time over the distributed storage. We, finally, describe how we deal with multi-node transactions.

4.1 Transaction Execution Model

As explained above, transactions are received by SNs, which handle them based on their access classes and the system status. To reach this goal, a SN node needs know where the data partitions are stored and, if ever such partitions are replicated on several XN, what is the less overloaded XN. The expected benefit is to minimize execution time. With this in mind, after receiving a transaction, T, modifying the partition p_i, a SN may process as follows:

- If τ_{p_i} is under its control, thus the SN reads metadata of p_i and assess where the T's latency will be the shortest regarding to its execution.
- else, it adds T in the pending list till it acquires τ_{p_i} as well as it may decide to forward T to another SN for shortening T's latency.

When SN_1 decides to forward a transaction T to SN_2, thus SN_2 will add T to its pending list and will handle all transactions when acquiring the corresponding token. In other words, transactions are grouped when SNs forward transactions to others in order to reduce waiting time.

4.2 Grouping Transactions Model

As stated in the previous section, a transaction can be routed once it arrives or pended and thus, delayed for an unbounded while. Nevertheless, one of the routing process goals is to minimize latency, thus, keeping a transaction into a pending list may induce the reverse effect, *i.e.,* to lengthen the response time. Let us consider the example of the ring depicted in Figure 2 and assume that T4 arrives immediately on SN_{k-1} after it releases the processing token that goes to SN_k. Thus, T4 is delayed at least for a complete round of the token. To avoid such a long waiting, SN_{k-1} may decide to forward T4 to one of the remaining SN that will surely receive the processing token before it. It is worth noticing that beside that forwarding transactions accelerates the routing process, it helps to parallelize the overall workload processing too. In fact, several SNs can send all transactions modifying p_i to the same SN that will get soon τ_{p_i}, hence, they have more availability to deal with transactions related to other

partitions. Transactions are grouped by using tokens either the processing one or the forwarding. Actually, we have three basic algorithms: a piggybacking-based, a ring-based and a time-based grouping model.

Piggybacking-Based Algorithm. This approach is defined to reduce transaction latency while minimizing the communication overhead for grouping transactions. Our solution is devised in such a way that we have as many processing tokens as number of partitions. Therefore, a SN may receive τ_{p_i} while it is using τ_{p_j}. In such a situation, the SN is constrained to let τ_{p_i} for another SN in order to avoid starvation. However, by releasing τ_{p_i}, the SN delays any transaction that was in its pending list and requiring p_i. To solve this problem, any transaction related to p_i is attached to τ_{p_i} and forwarded to the next SN. This strategy has the advantage to group transactions while sharing the processing tokens. Moreover, it ensures an efficient use of the tokens that see their roles multiplied.

Ring-Based Algorithm. The piggybacking strategy is tightly coupled with the synchronization process. That is, it helps to reduce latency only for those transactions that are attached to a SN receiving two tokens at a time while it requires only one. Hence, the improvement in the latency is biased and low since such situations are almost rare.

That is why we introduce a solution, namely the ring-based grouping method, which dissociates the synchronization process and the forwarding one. The ring-based model relies on a forwarding token and the ring to send or group transactions. To this end, only one forwarding token is shared by all SNs and for all partitions. Any SN receiving this token appends its pending transactions to it. The forwarding token circulates in a counter-clockwise for aggregating waiting transactions till it reaches a SN that already holds a processing token (let us say τ_{p_i}). Thus, the SN retrieves all transactions related to p_i and adds them to its own list for routing them and afterwards the forwarding token continues.

Moreover, the SN will not immediately release a processing token when the transactions end. Rather, the SN will keep the token as long as some transactions concerning p_i are waiting at that SN. This ensures that every forwarded transaction will eventually be executed. Intuitively, at any time a transaction is either at the SN which has the required processing token, either at another SN and waiting for the forwarded token to be caught, either being forwarded. We carefully manage the rare case where a transaction being forwarded would miss the processing token because of the token-crossing (i.e., the forwarded token jumps from SN_i to SN_j when the processing token jumps from SN_j to SN_i). We limit the maximum forwarding length to one round: a transaction is always dropped to a SN if it has been forwarded one entire round.

Time-Based Algorithm. The ring-based strategy performs well if the forwarding token quickly reaches its destination SN. This requires a small size ring with low latency, i.e., few SNs hosted in a local network. To overcome such locality and size constraint, we design a one-hop forwarding solution which relies

on a time-based estimation of the processing token location. This aims to enable transaction routing with larger rings covering a wider area.

With this strategy, each SN owns a forwarding token that can join a specified SN in one hop. Transactions are grouped based on token localization and the number of required partitions. To this end, a SN uses a timestamp for each token, $TS(\tau_{p_i})$, storing the last time it acquired τ_{p_i}. In this respect, whenever SN_i releases τ_{p_i} it updates $TS_i(\tau_{p_i})$. This timestamp is used to estimate when a SN will probably get again a token after releasing it. This estimation is calculated based on the number of SN and the moving average number of transactions per SN (let us say $\bar{T}$). Actually, if $\nu_{\bar{T}}$ is the time to route $\bar{T}$, thus, the idle time regarding a partition p_i of SN_j is: $idle(p_j^i) = (n-1) * \nu_{\bar{T}} + n * \lambda)$ where λ denotes the transfer latency of the token between two SNs and n the number of SNs in the ring. The idle time represents the time for a complete round of the token. We define the remaining waiting time, $rt(p_j^i)$ of SN_j considering that τ_{p_i} is hold in SN_k as:

$$rt(p_j^i) = idle(p_j^i) - [current\_time() - TS(\tau_{p_i})],$$

where *current_time()* is a call function, returning the current time of the system. Furthermore, when SN_j receives a transaction, T, requiring p_i, it assesses the remaining time $rt(p_j^i)$ to get τ_{p_i} and computes how many SN left, k, have to acquire the token before it by using the following formula:

$$k = \lceil rt(p_j^i)/(\nu_{\bar{T}} + \lambda) \rceil.$$

Once this value is known, SN_j decides to keep T if ever $k = 1$ or routes it to the next SN that will receive τ_{p_i} otherwise. In other words, a SN decides to keep an incoming transaction modifying p_i if ever it will be the one to get τ_{p_i} in the next step. If not, sending transaction to another SN helps to shorten the response time.

4.3 Transaction Ordering

With the grouping protocol described above, transactions may arrive from both clients (applications) and SNs side. That is, the order transactions arrive in the system globally may differ to the order a SN get them and process them. In fact, a SN routes transactions based on their arrival order to its pending list whatever their origin. Hence, T_i may be sent to SN_i before SN_j receives T_j and T_j being executed before T_i if ever it is forwarded to SN_j for the grouping reasons described in 4.1. Fortunately, executing transactions in an order different to their arrival is not so dramatic with respect to the class of social applications we consider. One reason may be that it is not worth to add new friends in Facebook with an specific order or to require an order for the enrollment to a Facebook event where the number of attendees is not limited. Furthermore, we ensure that a XN processes transactions with the same order sent by a SN and if ever a partition is replicated, all replicas will receive the same order. The motivation of applying the same order in all copies of a partition is to keep mutual consistency between replicas while ensuring an eventual consistency policy [11].

4.4 Managing the Processing Token

As pointed out early, processing tokens are used to synchronize concurrent transactions while avoiding starvation. We portray in the following subsections how we manage processing tokens to deal either with single or multi-partition transactions.

Single Partition Transaction. Our approach is devised to maintain a token per partition and each SN receiving a transaction has to add it into a pending list or to attach it to a forwarding token if the SN does not hold the corresponding processing token. That is, a SN waits until he gets a token for executing pending transactions in its queue. Basically, the token is hold by only one SN at a time, thus transactions requiring the same partition are executed in a sequential method and, therefore, no inconsistency can occur. Moreover, transactions are executed in a bounded time and starvation is almost avoided since: *i*) the number of transactions per SN is a finite number, thus, any SN will receive the processing token within a bounded interval (say the time for a complete round of the token as stated in the Section 4.2); *ii*) a SN cannot hold two processing tokens at a time, thus, tokens will circulate rapidly and distributed on different SNs.

Multi-partitions Transaction. When a transaction requires several partitions, acquiring all corresponding tokens may be somewhat tough and must be managed efficiently to avoid increasing the response time. To this end, we use the forwarding token not only to transfer transactions but to notify SNs to not use a particular token required for handling a multi-partition transaction. Moreover, each SN manages its own forwarding token and time-based grouping algorithm is used to forward transactions when needed. Actually, we detail the steps followed to handle a multi-partition transaction by considering Figure 3. Assume that SN_4 receives a multi-partition transaction requiring both p_i and p_j. Thus, it identifies the SNs (SN_6 and SN_2 respectively) holding the processing tokens τ_{p_i} (green diamond) and τ_{p_j} (blue diamond). Once this identification done, SN_4 sends a specific message, called an *inhibiting* message, by using its forwarding token to SN_6 and SN_2. An *inhibiting* message states that all successors of SN_6 and SN_2 must not use (or hold) τ_{p_i} and τ_{p_j} even if they have in their pending list some transactions related to p_i and p_j. That is, SN_4 inhibits successors of SN_6 and SN_2 and will be the next one to hold and allowed to use both tokens τ_{p_i} and τ_{p_j}. The intuition behind such a strategy is to give priority to multi-partition transactions that are somewhat less frequent. Moreover, when a successor is inhibited, it has to forward transactions in its pending list by using the grouping algorithm. In other words, the inhibiting process leads to group transactions and thus, to shorten their execution time as pointed out earlier.

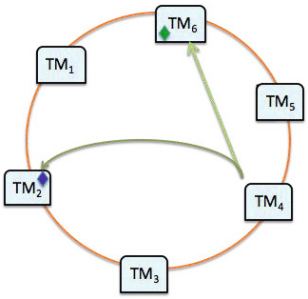


Fig. 3. Inquiring several processing tokens

5 Validation

In this, section we validate our approach through simulation by using SimJava [7], which is a toolkit (API Java) for building working models of complex systems. It is based on discrete events simulation kernel and includes facilities for representing simulation objects. We implement each of entities such as clients, SN nodes and XN nodes. Each entity is embedded into a thread and exchanges with others through events. To be as close as possible to a real system, each client that sends a query has to wait results before sending another one. To balance client requests over all SN nodes, clients use a round robin fashion to send a query to an SN node. Moreover, we implement TranspeerToken [10] a solution using a ring to route queries to a set of distributed databases.

The main objective of our experiments is to assess *String* performances, and afterwards, to compare them with TranspeerToken in terms of response time. To this end, experiments were conducted on an Intel duo core with 2 GB of RAM and 3.2 GHz running under Windows 7.

5.1 Evaluating Grouping Algorithms

Since our approach is mainly based on grouping transactions regarding to their access classes, we first conduct experiments to evaluate the benefits of our different strategies to group transactions (*i.e.,* piggybacking, ring-based and time-based algorithms). To this end, we set 10 SN nodes, 4 XN (*i.e.,* 4 partitions) and 10 clients making the workload. We let the system run till 500 transactions are processed. We compute for each algorithm, the percentage of transactions grouped based on data access classes, and we report results on Figure 4. One may see, that at least more than 40% of the transactions are grouped before their processing. This means, that our approach saves more than 40% of communication overhead compared to a solution that handles one transaction at a time. This gain demonstrates all the benefits of our scheduling layer that manages transactions in such a way that XN nodes may process transactions with group commits. Moreover, we observe that the time-based approach gathers 67,2% of

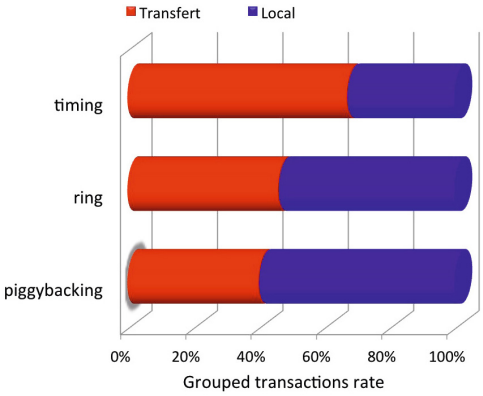


Fig. 4. Percentage of grouped transaction

transactions while the ring-based and the piggybacking ones do at most 46,2% and 40% respectively. The time-based algorithm overcomes the remaining algorithms because once it receives a transaction, it computes with more precision where to send it for minimizing the waiting time. Thus, when it receives several transactions within a short time window, the probability that it send all of them to the same SN nodes is very important as shown by results.

5.2 Analysing the Cost of the Grouping Algorithm

We focus on the communication cost of the distributed grouping algorithm. We estimate the number of messages that the grouping algorithm requires for each of the three steps. The collecting step operates in one round. Computing the groups is done locally at a SN, without sending any message. Dispatching the transactions operates in one round. Consequently, the entire grouping iteration requires 2*N messages. We now consider a whole run during which each application submit a sequence of several transactions. During such a run, the grouping algorithm iterates as long as there are some new incoming transactions to process. The iteration period may be adjusted (i.e., slowed down) to further reduce the communication cost of the grouping. Intuitively, it is worth waiting for most of the current transaction groups to end their execution, before running the next iteration of the grouping algorithm. This is especially interesting for groups containing hot (i.e. frequently demanded) partitions. Notice that the communication cost of the grouping algorithm is independent of the number of incoming transactions. It only depends on the ring size and the period at which the grouping algorithm executes. This makes the algorithm scalable to high workloads as we may see on next experiment.

For the piggybacking and ring-based algorithms, the grouping latency is increasing with the ring size. However, for the time-based algorithm, the grouping latency remains bounded, provided that the SNs forward the transactions, in a fixed (small) number of hops, to the SN which holds the token.

5.3 Absorbing Temporal Peak Loads

In this experiment we aim at showing that our solution may faces peak loads and therefore it is scalable. To this end, we set 10 SN, 10 XN and we vary the number of transactions from 1000 to 8000 in a very short time. We report on Figure 5 the response time as well as the workload increases. We observe as we argue previously that our solution faces efficiently peak loads since response time is almost constant when workload varies. The main raison is that even if thousands of transactions are sent, our scheduling layer gathers them in such a way that the overall routing time is reduce and hence, the overall response time. Moreover, we observe that all of our algorithms outperform TranspeerToken(TT) algorithm in terms of response time thanks to the grouping transactions step and group execution. We highlight too, that the ring and piggybacking give lower response time than the time-based approach. This is due to the fast grouping capability of the time-based approach which tends to group more transactions than the other approaches, before starting the transaction execution process. Therefore, this results shows that such fast grouping approach, although beneficial in the general case, might propagate the peak load to the grouping SN, with the drawback to lengthen the response time.

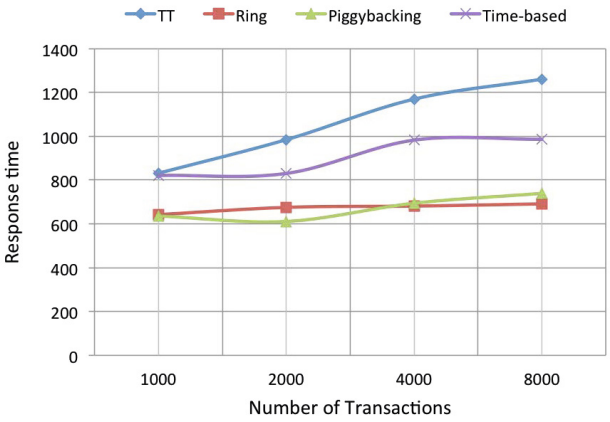


Fig. 5. Response time vs. worlkload

5.4 Impact of the Number of Tokens

Since there is one token per patition, the number of tokens grows as the number of partitions is incresing. This experiment is conducted to evaluate the impact of a growing number of tokens on the overall performances of *String*. To this end, we set a constant number of transactions (exactly 500 transactions), 10 SN nodes and we vary the number of XN from 5 to 35. We report of Figure 6 the results and we observe that adding more tokens increases the response time. The reason is that the overhead of managing "useless" processing tokens is increasing with the number of tokens. If too many processing tokens rotate into

the ring, the SNs frequently have to handle a token without using it, because either the SN is occupied with another token, or the token is not related with the pending transactions. This tends to slow down the forwarding token. Therefore a transaction will wait more time, on average, before being executed.

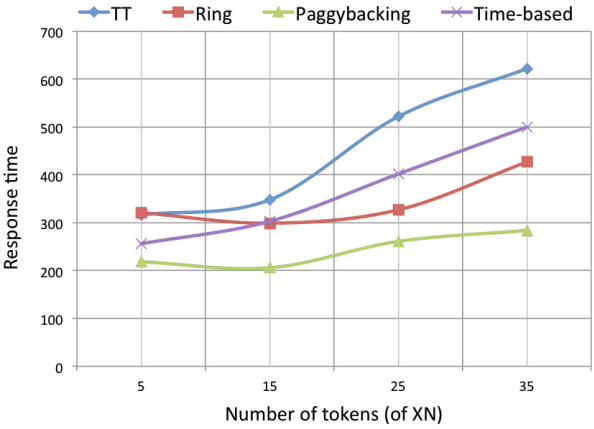


Fig. 6. Response time vs. number of XN (or number of tokens)

6 Conclusion

In this paper, we propose *String* a two steps approach, *i.e.,* a scheduling step followed by an execution step. *String* is a middleware-based solution devised over a ring, which ensures low latency when a peak load occurs. The main idea is to absorb the peak load, as early as possible, during the scheduling step, before processing the transactions. Actually, the transactions are ordered during the scheduling step in such a way that each transaction is followed or preceded by another one. Afterwards, each transaction is executed at the nodes storing the required data. A node-to-node coordination mechanism guaranties that the execution order remains identical for all the nodes involved. To reach our goal, we propose three algorithms we use during the scheduling step in order to group the transactions and to reduce the communication cost. We designed and simulated *String* using SimJava and we ran a series of experiments. Compared with some existing solutions, *String* shows interesting and promising results. Ongoing works are conducted to evaluate our solution in a cloud platform and to manage group transactions size for optimal execution.

References

1. Abadi, D.: Consistency tradeoffs in modern distributed database system design: Cap is only part of the story. IEEE Computer 45(2), 37–42 (2012)
2. Baker, J., et al.: Megastore: Providing scalable, highly available storage for interactive services. In: Conference on Innovative Data Systems Research (CIDR) (2011)

3. Carstoiu, D., Lepadatu, E., Gaspar, M.: Hbase - non sql database, performances evaluation. Int. Journal of Advancements in Computing Technology 2(5), 42–52 (2010)
4. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.: Bigtable: A distributed storage system for structured data. In: Symposium on Operating Systems Design and Implementation (OSDI), pp. 205–218 (2006)
5. O. Corporation. Oracle nosql database, 11g release 2
6. Helland, P., Sammer, H., Lyon, J., Carr, R., Garrett, P., Reuter, A.: Group commit timers and high volume transaction systems. In: International Workshop on High Performance Transaction Systems, pp. 301–329 (1989)
7. Howell, F., Mcnab, R.: simjava: A discrete event simulation library for java. In: International Conference on Web-Based Modeling and Simulation, pp. 51–56 (1998)
8. Kallman, R., Kimura, H., Natkins, J., Pavlo, A., Rasin, A., Zdonik, S., Jones, E.P.C., Madden, S., Stonebraker, M., Zhang, Y., Hugg, J., Abadi, D.J.: H-store: a high-performance, distributed main memory transaction processing system. Proc. VLDB Endow. 1(2), 1496–1499 (2008)
9. Lakshman, A., Malik, P.: Cassandra: a decentralized structured storage system. Operating Systems Review 44(2), 35–40 (2010)
10. Millet, L., Lorrillere, M., Arantes, L., Gançarski, S., Naacke, H., Sopena, J.: Facing peak loads in a p2p transaction system. In: Proceedings of the First Workshop on P2P and Dependability, P2P-Dep, pp. 1–7 (2012)
11. Sarr, I., Naacke, H., Gançarski, S.: Transpeer: Adaptive distributed transaction monitoring for web2.0 applications. In: ACM Symposium on Applied Computing, Dependable and Adaptive Distributed Systems Track (SAC DADS), pp. 423–430 (2010)
12. Silberstein, A., Chen, J., Lomax, D., McMillan, B., Mortazavi, M., Narayan, P.P.S., Ramakrishnan, R., Sears, R.: Pnuts in flight: Web-scale data serving at yahoo. IEEE Internet Computing 16(1), 13–23 (2012)
13. Thomson, A., Diamond, T., Weng, S.-C., Ren, K., Shao, P., Abadi, D.J.: Calvin: fast distributed transactions for partitioned database systems. In: ACM Int'l Conf. on Management of Data (SIGMOD), pp. 1–12 (2012)
14. Wei, Z., Pierre, G., Chi, C.-H.: Cloudtps: Scalable transactions for web applications in the cloud. IEEE T. on Services Computing 5(4), 525–539 (2012)

FOPA: A Final Object Pruning Algorithm to Efficiently Produce Skyline Points

Ana Alvarado, Oriana Baldizan, Marlene Goncalves, and Maria-Esther Vidal

Universidad Simón Bolívar, Venezuela
{aalvarado,obaldizan,mgoncalves,mvidal}@ldc.usb.ve

Abstract. We consider the problem of locating the best points in large multidimensional datasets. The goal is to efficiently generate all the points that meet a multi-objective query on data distributed in Vertically Partitioned Tables (VPTs). To compute the skyline on large VPTs, costly joins and comparisons may need to be executed, negatively impacting on the query execution time. We propose a new algorithm named FOPA (Final Object Pruning Algorithm) which is able to efficiently produce the whole set of skyline points and scales up to large datasets. FOPA relies on ordered VPTs, information on the values seen so far, and indices on the VPTs, to prune the space of dominated points and identify the skyline for large datasets in less time than state-of-the-art approaches. Empirically, we study the performance and scalability of FOPA in synthetic data and compare FOPA with existing approaches; our results suggest that FOPA outperforms existing solutions by up to two orders of magnitude.

1 Introduction

Nowadays, Web based infrastructures have been developed and allow large datasets to be published and accessed from any node of the Internet. Users more than ever can retrieve data that satisfies their requests by just searching or consulting any of the available sources of data. Although the democratization of the information provides the basis to discover properties and relationships that could not be identified years before, there are still applications where it is important to efficiently identify the best tuples that satisfy a query. That is, applications designed not only to meet soundness and completeness of the answers, but also to provide few relevant answers quickly. We address this problem and based on related work, we devise a solution to this ranking problem where tuples that best meet a given user request correspond to the skyline tuples.

A skyline is a set of tuples such that, none of them is better than the rest. Skyline techniques have gained great attention in the literature [7,9,17], and state-of-the-art approaches have focused on identifying the skyline tuples while the number of comparisons is minimized. In the database area, several techniques have been proposed to efficiently identify the best tuples in a skyline [3,11,19]; additionally, few approaches have considered this problem in the context of RDF, where data may be represented as VPTs [6]. Particularly, approaches proposed by Balke et al. [3] and Chen et al. [6] assume that the data is stored or distributed following a vertically partitioned table representation, i.e., for each data dimension or RDF property a, there exists a relation

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 334–348, 2013.

aR composed of two attributes, *Id* and *Value*; tuples are ordered according to the attribute *Value*. Further, a data structure is used to track the values of the objects seen so far. The algorithms work on iterations, where the best entries in each of the vertical tables are considered in one iteration. The goal of these algorithms is to minimize the number of comparisons between data dimensions to compute the skyline. First, the RDFSkyJoinWithFullHeader (RSJFH) algorithm proposed by Chen et al. [6], uses the data structure named header point, to record the worst values of the tuples explored in previous iterations; this information is used to guide the Pruning of tuples seen in future iterations. Second, the Improved Distributed Skyline Algorithm (IDSA) proposed by Balke et al. [3] was originally proposed for web sources, but can be naturally applied to VTPs. IDSA guides the search into the space of the most promising final object, i.e., a final object is the one that has been considered in all the VPTs; once the final object is found IDSA can ensure that a superset of the skyline has been found. Although experimental studies reported in the literature[3,6] suggest that RSJFH and IDSA are efficient, both algorithms do not scale up to large datasets and suffer of the following drawbacks: *i*) RSJFH may produce incomplete results, and *ii*) IDSA performs a large number of comparisons.

To overcome limitations of RSJFH and IDSA, we define a new algorithm named FOPA (Final Object Pruning Algorithm). FOPA also assumes that elements or tuples in a dataset are characterized by multi-dimensions and are stored following the vertical partition approach, ordered and indexed by two indices. Additionally, FOPA maintains information about the last values seen so far, and exploits this information to prune within the group of objects read in a given iteration, those objects that will not be part of the skyline. This allows FOPA to compute the skyline set as soon as it finds the final object or a dimension has been completely scanned, not incurring in any further comparisons. Thus, FOPA can ensure completeness while the number of comparisons and data access is reduced. We empirically studied the properties of FOPA with respect to state-of-the-art approaches; our experimental results suggest that techniques implemented by FOPA allow the reduction of execution time by up to two orders of magnitude in synthetic data produced using the data generator proposed by Borzsonyi et al. [5]. To summarize, the main contributions of this paper are:

- the Final Object Pruning Algorithm which is able to efficiently identify the tuples that best meet a user multi-objective request; the approach is complete and scales up to large datasets, while the number of comparisons are reduced; and
- an extensive empirical study where performance and quality of the proposed algorithm is evaluated in terms of RSJFH[3] and IDSA[6]; results suggest that the proposed approach overcomes these solutions.

This paper is composed of five additional sections. Section 2 illustrates a motivating example and describes the main properties of RSJFH[3] and IDSA[6]; section 3 formalizes FOPA. Section 4 presents an experimental study where we report on the performance and the quality of FOPA. Section 5 summarizes existing state-of-the-art approaches. Finally, we conclude in section 6 with an outlook to future work.

2 Motivating Example

Suppose a tourist is interested in visiting the nearest restaurants to her location that serve the best local food, have the best decoration and are the most popular; quality, decor and popularity are equally important for her. In order to have a good selection of restaurants to visit, the tourist wants to review the restaurants that best meet her conditions. Following our tourist's criteria, a restaurant will be preferred if there is no other restaurant with higher quality, decor and popularity. Figure 1(a) presents a table with the nearest restaurants that serve local food. Formally, a set of preferred restaurants is composed of restaurants that are non-dominated by any other restaurant in terms of the former criteria; this set of restaurants is called skyline. A restaurant A dominates a restaurant B, if B has better values in quality, decor and popularity than A, e.g., restaurant 4 is dominated by restaurant 1. Additionally, restaurants 1, 2, 3, and 11, are the non-dominated ones, i.e., there is no other restaurant r in the table of Figure 1(a) with values better than restaurants 1, 2, 3 and 11 in these three attributes. Notice that a weight or a score function cannot be assigned because all the criteria are equally important and relevant. The problem of computing the skyline is polynomial in the size of the dataset, and the goal of state-of-the-art algorithms is to compute the set of skyline points without having to perform a polynomial number of comparisons[9,11,17].

Id	Quality	Decor	Popularity
1	10	4	2
2	2	10	3
3	6	2	10
4	9	3	2
5	1	10	3
6	1	2	9
7	9	4	1
8	2	8	2
9	6	1	8
10	8	4	1
11	1	9	8

(a) Table of Near Restaurants

Id	QualityValue
1	10
4	9
7	9
10	8
9	6
3	6
8	2
2	2
5	1
11	1
6	1

(b) QualityR

Id	DecorValue
2	10
5	10
11	9
8	8
7	4
10	4
1	4
4	3
6	2
3	2
9	1

(c) DecorR

Id	PopularityValue
3	10
6	9
9	8
11	8
5	3
2	3
8	2
4	2
1	2
10	1
7	1

(d) PopuR

Fig. 1. (a) Table of Near Restaurants and its Vertically Partitioned Table Representation (b), (c) and (d)

Recently, the vertically partitioning representation has gained attention in the literature not only for efficiently representing RDF data, but also for providing a compact implementation of relational tables on top of column-oriented databases [1]. Formally, in a vertically partitioned table representation, for each attribute of a relational table or RDF property a, there exists a relation aR composed of two attributes, *Id* that represents the identifier of a tuple or the URI of an RDF resource, and *Value* that corresponds to the value of the attribute or property assigned to the entity with identifier *Id*; additionally, tuples can be ordered according to the attribute *Value*. Following this idea, tuples presented in Table of Figure 1(a) are modeled as three vertically partitioned tables, as shown in Figures 1(b), (c) and (d). Data in tables `QualityR`, `DecorR` and `PopuR`

are stored in descending order. Vertically partitioned tables can support heterogeneous values, reduce the number of nulls, speed up the execution of merge join operators, and reduce the number of I/Os. Because skyline approaches can be widely benefited from exploiting advantages of the vertically partitioned tables, we focus on algorithms that rely their ranking process on sorted binary tables that represent relationships of an entity with each of its attributes or properties. We consider two RSJFH [6] and IDSA [3]. Both algorithms implement a two-fold approach where in the first phase a set of skyline candidates is identified, and then, in the second phase, dominated candidates are eliminated by performing an existing skyline algorithm. To compute the set of skyline candidates, these two algorithms assume that the data is stored following a Vertically Partitioned Table (VPT) representation. Additionally, indices are kept on top of these tables to provide direct and sequential access, and a data structure is used to track the values of the objects seen so far. The algorithms work on iterations, where the best entries in each of the VPTs are considered in one iteration. Entries from different VPTs are joined to retrieve the values of all the attributes of an entry; these tuples are named *joined tuples*. The goal of these algorithms is to minimize the number of comparisons to compute the skyline candidates.

The RDFSkyJoinWithFullHeader (RSJFH) algorithm proposed by Chen et al. [6], uses the data structure named header point, to record the worst values of the tuples explored previously; this information is used to prune tuples in future iterations. A joined tuple is pruned if it is worse than the header point in at least one dimension. RSJFH finalizes the first phase when either of the following conditions holds: *i*) the header point is not updated or *ii*) at least one of the tables is explored completely. We illustrate the main drawbacks of RSJFH in our running example. First, the RSJFH algorithm reads the tuples in each dimension and performs a hash join to retrieve the joined tuples: t_1=<1, 10, 4, **2**>, t_2=<2, **2**, 10, 3> and t_3=<3, 6, **2**, 10>[1]. These first tuples will create the header point (H) with the worst values in each dimension, resulting in H=<2, 2, 2>, i.e., bold values in t_1, t_2 and t_3 correspond to the entries in H; additionally, t_1, t_2 and t_3 are the tuples associated with the header. Next, the algorithm reads the following tuple in `QualityR` and the joined tuple obtained is t_4=<4, 9, 3, 2>. Compared with H, t_4 is not pruned, because it has the same value in `PopularityValue` as H; thus, t_4 is included in the set of header tuples. Then, RSJFH reads the next tuple in `DecorR`, and performs a hash join to build t_5=<5, **1**, 10, 3>. Compared with H, t_5 is pruned because it is worse than H in at least one dimension (`QualityValue`). Since, t_5 was pruned, the algorithm continues reading the same table and retrieves the joined tuple t_{11}=<11, **1**, 9, 8> which is also pruned. Nevertheless, t_{11} is a skyline tuple, illustrating why RSJFH can produce incomplete results. The algorithm reads the next tuple and performs the hash join to obtain t_8=<8, 2, 8, 2> which is not pruned and is included in the set of header tuples. Then, RSJFH considers `PopuR` table, reads the next tuple and obtains the joined tuple t_6=<6, **1**, 2, 9> which is pruned. This process is repeated until the algorithm reaches the end of `PopuR`; one of the stop conditions. Once this phase is finished, seen and unpruned tuples correspond to the candidates, and they are passed to the BNL algorithm to compute the skyline.

[1] For each t_i, arguments correspond to *Id*, *QualityValue*, *DecorValue* and *PopularityValue*, respectively.

The Improved Distributed Skyline Algorithm (IDSA) proposed by Balke et al. [3] performs two types of accesses: sorted and random. Sorted accesses follow the sequential order of the VPTs, while random accesses are performed to build the joined tuples. Similarly to RSJFH, IDSA maintains a data structure to record last seen values among the already explored tuples, we named this structure the header object. Also, IDSA distinguishes one object: the most promising final object, which is the one most likely to be seen in all the dimensions following sorted accesses. Additionally, IDSA keeps two values c_{val} and f_{val} related with the header object; c_{val} and f_{val} are computed as the sum of the differences between the values of the header object and the current joined tuple, and between the header object and the most promising final object, respectively. Initially, the most promising final object is the first joined tuple, and this is updated with the current joined tuple only when c_{val} is smaller than f_{val}. In case both c_{val} and f_{val} hold the same value, the sum of their respective objects' values is used to determine which will be the most promising object. IDSA finalizes the first phase when all the values of the most promising final object were seen following sorted accesses. Once the final object is found, IDSA can ensure that a superset of the skyline has been found, and a post-processing step is fired to discard the tuples in this superset that are dominated.

IDSA initializes the header object with the *maximal* values from the VPTs, i.e., H=<10,10,10>. Then, IDSA reads the first tuple t_1 of `QualityR` and performs random accesses to find the rest of its values and build a joined tuple. Because t_1 has value 10 in `QualityValue`, the header object is not updated. Next, IDSA sums up the difference between values of t_1 and the header object values, and computes c_{val}, i.e., c_{val} is 14. Thus the tuple t_1 becomes the most promising final object and therefore, f_{val} is updated to 14. IDSA considers one of the tables where t_1 has not been found following a sorted access, for example, `DecorR`. Thus, IDSA reads the next tuple and performs the random accesses to obtain t_2=<2, 2, 10, 3>. Because H and t_2 have the same value in `DecorValue`, H is not updated. Then, IDSA adds up the difference between t_2 and the header object, and it assigns 15 to c_{val}. The final object is not updated because c_{val} is greater than f_{val}. IDSA continues on `DecorR` because t_1 has not been seen yet in this VPT. IDSA reads t_5=<5, 1, 10, 3>, and the header object is not updated. Then, IDSA adds up the difference between t_5 and the header object, c_{val} is 16, which is greater than f_{val} . IDSA continues and reads the next tuple t_{11}=<11, 1, 9, 8> and the header object is updated to H=<10, 9, 10>; thus, f_{val} and c_{val} are updated to 13 and 11, respectively. Because f_{val} is greater than c_{val}, the most promising final object is now t_{11}. Now, IDSA considers one of the tables where t_{11} has not been seen, e.g., `QualityR`. IDSA reads the next tuple and performs the random accesses to obtain t_4=<4, 9, 3, 2>, and the header object is updated to H=<9, 9, 10>, f_{val} is 10, and c_{val} is 14. This process is repeated until the algorithm finds t_{11} in the rest of the tables.

As shown in the previous example, both RSJFH and IDSA suffer of the following drawbacks: *i*) RSJFH produces incomplete results when skyline results have at least one value worse than the one registered for the same dimension in the header, i.e., t_{11}, and *ii*) IDSA performs a large number of comparisons and joins, when the most promising final object is comprised of a value that is close to the worst value of one of the VPTs, i.e., t_{11} was composed of the worst value in `QualityR` and IDSA had to read this table completely following the sorted access. To overcome these limitations and improve

efficiency, we propose the algorithm FOPA which combines the best properties of the algorithms RSJFH and IDSA. FOPA registers enough information about the seen values to prune dominated tuples, and to select the VPTs where the most promising object will be read; thus, the comparisons and joins can be reduced while the solution is complete.

3 Our Approach

Similarly to RSJFH and IDSA, FOPA also assumes that data is stored following a vertically partitioned table representation; additionally, each VPT is indexed by two indices, one on the attribute *tupleId* and the other on *aValue*; tuples are ordered in *aR* based on the values of a. Also, FOPA keeps the last score values of each dimension in the header object, to guide the search for the most promising final object. Unlike IDSA and RSJFH, FOPA identifies the skyline tuples in only one phase. In one iteration, FOPA reads all the tuples containing the same value in a given VPT; this may result in multiple tuples being read at the same time which are compared among themselves to discard the ones that are dominated. This set of non-dominated tuples will be compared against the skyline tuples found in the corresponding VPT, i.e., the skyline objects that have been seen previously by performing sorted accesses in the current VPT.

Similar to IDSA, FOPA computes c_{val} and f_{val}, and initializes the most promising final object with one of the first joined tuples that is not dominated among the group it was read with. Every time a newly joined tuple is neither discarded nor dominated, c_{val} is computed for it, and f_{val} is recalculated. Then, the most promising final object will be updated whenever c_{val} is lower than f_{val}. FOPA finalizes when all the values of the most promising final object are seen following sorted accesses, or when any of the tables is completely explored. A sketch of FOPA is presented in Algorithm 1.

As a first step, FOPA initializes the header object with the *best* values from the VPTs, i.e., H=<10,10,10>. Then, FOPA reads the first tuple t_1 of `QualityR` and performs random accesses to find the rest of its values and build a joined tuple (Line 3 Algorithm 1). Object t_1 is added to the set of seen objects and since it is the first iteration, t_1 becomes the most promising final object which we will call fid from now on (Line 9 Algorithm 1). Because t_1 has value 10 in `QualityValue`, the header object is not updated. Since no other objects have been read so far, t_1 is not pruned by FOPA; it is added both to the set of seen objects P, and to the skyline set S (Line 11 Algorithm 1). Next, FOPA sums up the difference between values of t_1 and the header object values, and computes f_{val}, i.e., f_{val} is 14, which in this first iteration will be the same as c_{val} (Line 19 Algorithm 1). FOPA considers one of the tables where t_1 has not been found following a sorted access, for example, `DecorR` (Line 27 Algorithm 1). Thus, FOPA reads the tuples with the same `DecorValue` and performs the random accesses to obtain t_2=<2, 2, 10, 3> and t_5=<5, 1, 10, 3> (Line 3 Algorithm 1). Because t_5 is dominated by t_2, t_5 is discarded after adding it to the set of seen object P (Line 5 Algorithm 1). Object t_2 is now part of the set of non-dominated tuples S (Line 11 Algorithm 1), and also part of the set P. FOPA assigns 15 to c_{val} and the fid remains the same because c_{val} is greater than f_{val} (Line 19 Algorithm 1). FOPA continues in `DecorR`, and it reads the next tuple t_{11}=<11, 1, 9, 8> (Line 3 Algorithm 1). The tuple t_{11}, is not dominated or pruned by any other tuple and thus, it is added to the skyline

Algorithm 1. FOPA Algorithm

1: Initialize two data structures $S, P \leftarrow \emptyset$ as the set of skyline objects, and seen objects, respectively. Initialize n data structures one for each dimension, $k1$,..., $kn \leftarrow \emptyset$ containing records with an identifier. Initialize a record fid and initialize two real values $c_{val}, f_{val} \leftarrow 0$. Initialize the header H with n real values $h1, ..., hn$ obtained from the best values in each dimension. Initialize $i \leftarrow 1$, and $phase2 \leftarrow \emptyset$, a lists of records.
2: **while** fid is not in every $ki \wedge$ unseen elements exist in the current list D_i **do**
3: Get all the records which hold the same value for dimension i, by performing sorted access on list D_i.
4: Ignore all records that already exist in P and not in S.
5: Discard all objects that are dominated among each other in this group and add them to P, add the remaining non-dominated objects into list $phase2$.
6: **while** $phase2 \neq \emptyset$ **do**
7: Read an object $objp$ from $phase2$ and remove it from the list
8: **if** $objp$ has not been read already **then**
9: Add $objp$ to P, and if this is the first iteration, set fid to $objp$.
10: **if** $objp$ is not pruned by the existing skylines in dimension i **then**
11: Add $objp$ to S.
12: **end if**
13: **end if**
14: Add $objp$ to list ki of seen objects in dimension i.
15: **if** $objp \in S$ **then**
16: Update hi value in header with the value on the i-th dimension in $objp$.
17: Compute the sum of differences between $objp$'s absolute score values, and the values in the header and assign this value to c_{val}.
18: Compute the sum of differences between the fid's absolute score values, and the values in the header and assign this value to f_{val}.
19: **if** $c_{val} < f_{val}$ **then**
20: Replace the current fid with $objp$.
21: **end if**
22: **end if**
23: **end while**
24: **if** fid object is in every ki **then**
25: Exit While.
26: **else**
27: Set i from D_i to a dimension in which fid has not been seen.
28: **end if**
29: **end while**
30: Output S as the set of skylines.

set S and to the set of seen objects P (Lines 9-11 Algorithm 1). FOPA updates the header to H=<10, 9, 10>; and so, f_{val} and c_{val} are updated to 13 and 11, respectively (Lines 16-18 Algorithm 1). Because f_{val} is greater than c_{val}, the new fid is now t_{11} (Line 19 Algorithm 1). Now, FOPA considers one of the tables where t_{11} has not been seen, e.g., `QualityR` (Line 27 Algorithm 1). FOPA reads the next tuples and performs the random accesses to obtain t_4=<4, 9, 3, 2> and t_7=<7,9,4,1>. The set of non-dominated tuples is t_4 and t_7 and they are both added to the set of seen tuples P (Lines

5-9 Algorithm 1); however, both tuples are dominated by skyline t_1, then, neither is added to the skyline set (Line 10 Algorithm 1). Next, FOPA reads t_{10} which after being added to set P, is dominated by t_1 (Line 10 Algorithm 1). Later, the algorithm considers t_9 and t_3; the first one is dominated by t_3 so it is added to the set P and then discarded (Line 5 Algorithm 1). After adding t_3 to the set P of seen objects (Line 9 Algorithm 1), it is determined that t_3 is a non-dominated tuple and is therefore added to the skyline set S (Line 11 Algorithm 1); and c_{val} and f_{val} are updated with 7 and 7, respectively (Lines 17,18 Algorithm 1). Then FOPA reads t_8 and t_2; t_8 is discarded because it is dominated by t_2, and added to the set P of seen objects. Because t_2 is already in sets P and S, fid is not updated. Finally, FOPA accesses tuples t_5, t_{11} and t_6. The tuple t_5 is ignored since it has already been read and is not a part of the skyline (Line 4 Algorithm 1). Then, the algorithm prunes t_6, because t_3 dominates it. Since the end of `QualityR` has been reached, FOPA outputs t_1, t_2, t_3 and t_{11} as the skyline and the algorithm concludes (Lines 2,30 Algorithm 1). Note that FOPA reduces the number of comparisons and speeds up the time required to completely read one VPT. This is because FOPA considers groups of tuples that have the same value in one of the dimensions, and compares among themselves to select the non-dominated ones; then, the resulting tuples are compared with the skyline tuples. Table 2 reports on the number of comparisons performed by RSJFH, IDSA, and FOPA. We can observe that even in this simple example, FOPA reduces the number of operations while the answer is complete.

Algorithm	Comparisons	Joins	Accesses	Skyline Size
RSJFH	25	17	17	**3**
IDSA	207	11	19	4
FOPA	**22**	13	**14**	**4**

Fig. 2. Number of Comparisons, Joins, Sorted and Random Accesses, and Skyline Size

Theorem 1 (FOPA Pruning Correctness). *Every discarded or pruned tuple is not part of the skyline set.*

Proof. Let *P*=$< p_1, \ldots, p_n >$ be a skyline tuple, *newly* read in dimension *i*, that is pruned. There are two possible situations in which *P* could have been pruned; *i*) either a tuple *Q*=$< q_1, \ldots, q_n >$, which holds the same value as *P* in dimension *i*, dominates *P* or *ii*) a skyline tuple *T*=$< t_1, \ldots, t_n >$, which was previously found in dimension *i*, dominates *P*. In the first case, if *Q* dominates *P*, then it has equal or better values than *P*, and at least one value strictly better; however, this contradicts the definition of skyline, and therefore *P* could not be part of the skyline set. In the second case, we know *T* is strictly better in dimension *i* than *P*, hence the rest of its values should be equal or better than *P*, in order to prune it, which contradicts the skyline definition.

Theorem 2 (FOPA Pruning Completeness). *Every read tuple that is not part of the skyline set is discarded or pruned.*

Proof. Let *P*=$< p_1, \ldots, p_n >$ be a newly read tuple that is not member of the skyline, then there exists a skyline point *Q*=$< q_1, \ldots, q_n >$ which dominates it. In order for

P to be dominated by Q, according the definition of skyline, Q must be at least equal in $n-1$ dimensions and strictly better than P in one dimension. Thus, there are two possible situations that may develop; *i*) either Q has been read before P, both in the same dimension, and therefore, P will be pruned, or *ii*) Q and P have been read in the same group in one dimension, in which case since Q dominates P, P will be discarded.

4 Experimental Study

We empirically analyze and report on the performance and quality of FOPA with respect to the algorithms RSJFH[3] and IDSA[6].

Table 1. Skyline Cardinality Per Dataset and Data Distribution. Large Skyline produced by complex queries are highlighted in **bold**.

Data Distribution	Dataset Cardinality	Skyline Cardinality Ranges
Anti Correlated	10,000	50-2,995
	50,000	54-9,189
	100,000	**77- 13,111**
	1,000,000	11-2,653
	2,000,000	13-1,735
	3,000,000	15-2,008
	4,000,000	16-2,122
Correlated	10,000	105-2,969
	50,000	150-7,250
	100,000	**201-10,174**
	1,000,000	15-2,414
	2,000,000	18-2,256
	3,000,000	17-2,652
	4,000,000	15-2,680
Independent	10,000	50-4,873
	50,000	59-15,660
	100,000	**70-23,791**
	1,000,000	11-1,696
	2,000,000	18-2,326
	3,000,000	17-2,625
	4,000,000	18-2,720

Datasets and Queries: we considered synthetic datasets generated by the benchmark generator implemented by Borzsonyi et al. [5]. We generated data with three different distributions: correlated, anti-correlated and independent (uniform). The values in each dimension, range from 0.0 to 1.0. Additionally, for each distribution seven datasets were generated with 10,000, 50,000, 100,000, 1M, 2M, 3M and 4M tuples, respectively. For the 10,000, 50,000 and 100,000 datasets, tuples are characterized by an identifier and 3, 5 and 10 dimensions. For the 1M, 2M, 3M and 4M datasets, tuples are characterized by an identifier and 2, 3, 4 and 5 dimensions. Thus, experiments were conducted on 75 different datasets. The performed queries where characterized by the MIN and MAX directives, as anti-correlated queries i.e., alternated MIN MAX directives for each attribute of the multi-objective queries.

Evaluation Metrics: we measure performance as *query execution time* and *throughput*. The former is computed as the *elapsed time* between a query submission and the generation of the skyline points; this is measured by the *nanoTime()* method from the Java's

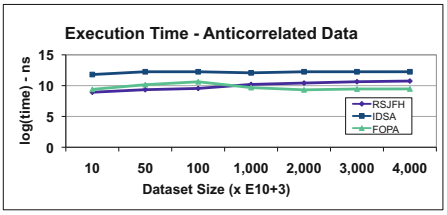
System Class. The latter reflects the number of skyline tuples that an algorithm can produce per second; this is computed as the ratio of the number of skyline tuples generated by an algorithm to the elapsed time in seconds. Effectiveness is reported as *answer completeness*, and *correlation* between *number of pruning* versus *answer completeness*. Answer completeness corresponds to the percentage of the total skyline produced by an algorithm, while the *correlation* reflects when an algorithm prunes true skyline candidates; points with X's values close to 0% indicate that skyline points were wrongly pruned, while points with X's values close to 100% show that prunes were effective and did not affect the completeness of the answer.

Implementations: RSJFH, IDSA and FOPA were all implemented in Java (64-bit JDK version 1.7.0 17); we followed the vertical partitioning approach to represent data in a relational database in Oracle 10g and accessed through the JDBC API. Experiments were executed on a Sun Fire V440 equipped with two UltraSPARC IIIi processors running at 1.593 GHZ with 12GB RAM and 2 disks HITACHI HDS7225 and SEAGATE ST32500 of 250 GB each running on SunOs 5.10 (Solaris 10). In IDSA's approach, we have taken into consideration the possibility of requiring different conditions set on the dimensions consulted in a query e.g. minimum and maximum. Therefore, when calculating the sum of differences between all the values of an object and the last score value seen in a given iteration, the absolute value of each addend is used, rather than only the original addend. Timeout is of 30 minutes.

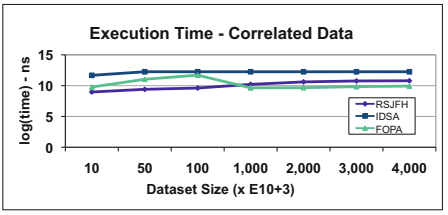
4.1 FOPA Performance

We study the performance of FOPA with respect to the algorithms RSJFH [3] and IDSA [6] on the synthetic datasets described in Table 1; note that the skyline cardinalities differ by up to two orders of magnitude. The effects of the cardinality of the dataset, the data distribution, and the size of the skyline on the performance of these algorithms are studied; performance is measured in terms of execution time and throughput. Figures 3 (a), (b) and (c) report on the execution time in logarithmic scale. We can observe that FOPA and RSJFH consume less time than IDSA; in fact, IDSA timed out in a great number of the queries with more than 3 dimensions. FOPA's performance seems to be more affected by the complexity of the queries (up to 10 dimensions datasets with 10,000; 50,000; 100,000) than for the size of the dataset (queries with 2, 3, 4 and 5 dimensions for datasets of 1M, 2M, 3M and 4M); note that FOPA's execution time can be increased by up to two orders of magnitude when the dataset grows into 100,000 tuples and the queries are complex. Nevertheless, FOPA's execution time remains almost stable for queries of up to 5 dimensions and large datasets; this indicates that FOPA can scale up to large datasets whenever complexity remains in certain range. Contrary, RSJFH's performance does not seem to be affected by either the complexity of queries or dataset size; however, the completeness of the answer considerably decreases as we can observed in Figure 4. Additionally, Figure 3(d) illustrates the performance of RSJFH, IDSA and FOPA in terms of the number of skyline tuples that each one can produce per second. In all cases, FOPA produces up to one order of magnitude more skyline tuples per second than RSJFH and IDSA do. As explained before, throughput is more affected by the complexity of the queries executed against datasets of size 10,000;

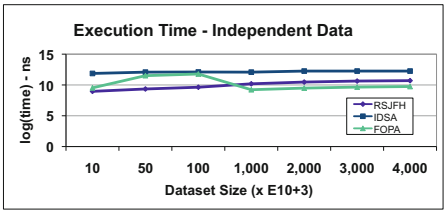
50,000; and 100,000 than by the size of the datasets. Thus, we can observe that FOPA's throughput values decrease rapidly for datasets of 10,000; 50,000; and 100,000 where complex queries are executed. IDSA's and RSJFH's throughput curves also decline but the slopes are smaller than FOPA's throughput slope.



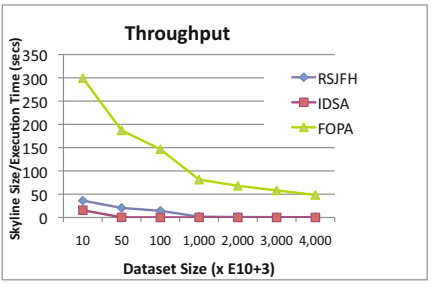
(a) Execution Time-AntiCorrelated Data (ns)



(b) Execution Time-Correlated Data (ns)



(c) Execution Time-Independent Data (ns)



(d) Throughput (Number of Skyline Tuples Per Second)

Fig. 3. Performance measured in terms of Execution Time (logarithmic scale) and Throughput. Queries against Datasets with 10,000; 50,000 and 100,000 have 3, 5, and 10 dimensions. Queries on Datasets of 1M, 2M, 3M and 4M are of 2, 3, 4 and 5 dimensions. Timeout of 30 minutes.

4.2 FOPA Effectiveness

We measure effectiveness in terms of completeness of the answer and the correlation between the number of prunings and the completeness of the answer. Figure 4 presents the percentage of answer completeness for queries of dimensions 2, 3, 4, and 5, in datasets of 1M, 2M, 3M and 4M; three data distributions are considered. FOPA and IDSA are complete, and they always reach an answer completeness of 100%; similar behavior is exhibited by RSJFH for queries of 2 dimensions and datasets of any size or data distribution. Nevertheless, because RSJFH aggressively prunes seen objects, it can discard a large number of skyline candidates and is able to produce less than 3% of the skyline tuples for queries of 5 or more dimensions. This behavior can be

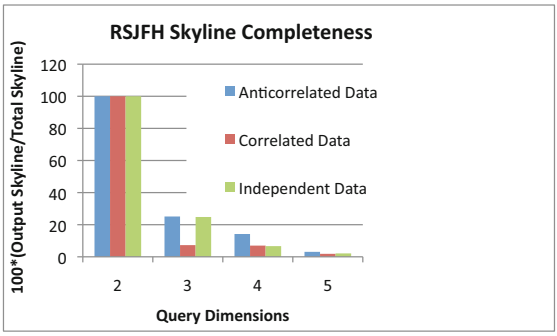


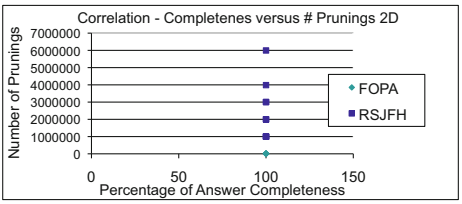
Fig. 4. RSJFH Percentage of Answer Completeness. IDSA and FOPA produce all the skyline tuples, and are not reported in the plot for legibility.

also observed in Figures 5 (a), (b), (c), and (d). It is important to observe that FOPA's pruning strategy always discards false skyline tuples (green points in Figures5 (a), (b), (c), and (d) have X's values of 100%). However, RSJFH only exhibits this behavior for queries of 2 dimensions, and the effectiveness of the pruning strategy deteriorates until the point that for queries of 5 dimensions all the blue points have X's values close to 0; this indicates that most of the discarded tuples were true skylines.

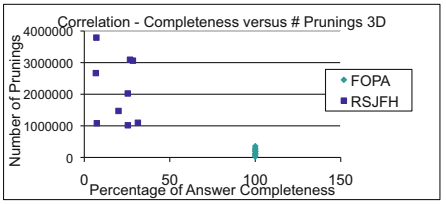
4.3 Discussion

We conducted an empirical evaluation on a large variety of datasets and queries: 75 datasets of different sizes, data distributions, and dimensions were generated; additionally, queries of different dimensions that produced skylines of a wide range of cardinalities were evaluated. The proposed algorithm was compared with RSJFH and IDSA. Observed experimental results allow us to conclude the following:

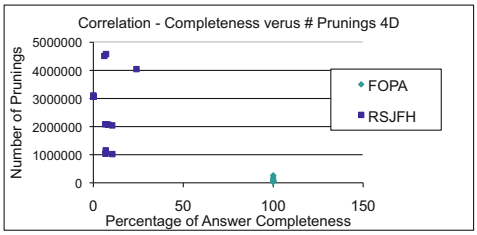
- IDSA and FOPA are able to produce complete answers independently of the sizes of the datasets, the number of dimensions, data distributions or skyline sizes.
- IDSA does not implement any pruning technique, so it requires to traverse the VPT's until a final object is completely seen. In case of large skylines, IDSA may time out before producing any skyline tuple.
- RSJFH implements an aggressive pruning strategy that effectively prunes false skyline candidates when queries are of 2 dimensions. Nevertheless, the quality of this pruning strategy deteriorates as the complexity of the queries increases, and almost all the tuples in the skyline can be pruned.
- FOPA exhibits a good performance in large datasets and the pruning strategy discards only seen points that do not belong to the skyline. However, FOPA's performance can be affected if queries are comprised of 10 or more dimensions, and the skylines are large.



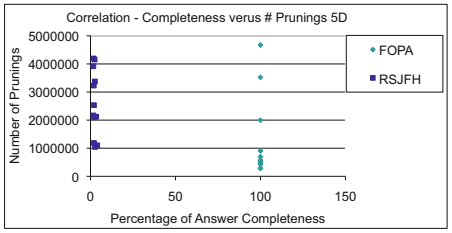
(a) Correlation 2 Dimensions



(b) Correlation 3 Dimensions



(c) Correlation 4 Dimensions



(d) Correlation 5 Dimensions

Fig. 5. Quality measured in Correlation Between Percentage of Answer Completeness Versus Number of Prunings

5 Related Work

Skyline [5] approaches have been defined in the context of databases to distinguish the best tuples that satisfy a given ranking condition. A skyline-based technique identifies a partially ordered set of services whose order is induced by criteria composed of conditions on equally important parameters. The problem of efficiently computing the skyline has been extensively studied in the literature [2,3,5,7,9,12,13,14,15,16].

Bentley et. al. [4] propose the first skyline algorithm, referred to as the maximum vector problem and it is based on the divide & conquer principle. Progress has been made as of recent on how to compute efficiently such queries in a relational system and over large datasets. Block-Nested-Loops (BNL) [5], Sort-Filter-Skyline (SFS) [8] and LESS (Linear Elimination Sort for Skyline) [10] are three algorithms that identify the skyline by scanning the whole dataset. On the other hand, progressive (or online) algorithms for computing skyline have been introduced: Tan et al. propose the NN (Nearest Neighbor) [18] algorithm and BBS (Branch-and-Bound Skyline) [14,12]. A progressive algorithm returns the first results without having to read the entire input and produces more results during execution time. Although these strategies could be used to implement our approach, they could be inefficient because they may perform a number of non-necessary comparisons or generate an incomplete answer.

In order to process skyline queries against web data sources, efficient algorithms have been designed considering sequential and random accesses. Each data source contains

object identifiers and their scores. A sequential access retrieves an object from a sorted data source while a random access returns the score from a given object identifier. Balke and Guntzer proposed the Basic Distributed Skyline (BDS) [2], a two-fold algorithm, which builds the skyline in two phases; first a superset is constructed, and then, dominated points are eliminated. BDS performs sorted access in a round robin fashion on the data set in order to compute a Skyline superset. BDS stops in the first phase when it completely sees an object. A second algorithm known as IDSA is presented by Balke et al. [3]; in contrast to BDS, it compares all the seen objects once it finds that some seen object dominates a most promising object that is updated constantly; additionally, IDSA maintains a data structure to record last seen values among the already explored tuples, we named this structure the header object. IDSA scans the same objects than BDS but constructs the final object. The final object is comprised of values that have been seen in all the VPTs by performing sequential access. For each seen object, IDSA performs random access for retrieving unseen values and compares pair-wise the seen objects versus the most promising object and the header object. Selke and Balke [15] proposed a trie-based indexing technique called SkyMap, able to address all problems for skyline queries in realistic applications, in order to achieve faster access, superior scalability, and low costs for maintenance due to data updates. This technique uses a tree data structure that adopts a ”data-independent disjoint space partitioning” and navigation based in binary strings. To improve efficiency they rely on inexpensive bitwise operations to explore and create the tree. Additionally, SkyMap avoids re-balancing by storing the data only in the leaf nodes and by using the Z-address concept. Recently, Chen et al. [6] propose the RSJFH algorithm to compute the skyline on RDF documents that have been represented as VTPs; this algorithm is guided by a header point that summarizes the boundaries of the visited space. The header point records the worst values of the tuples explored previously; this information is used to guide the pruning of tuples in future iterations. A joined tuple is pruned if it is worse than the header point in at least one dimension. Although experimental studies reported in the literature[3,6] suggest that RSJFH and IDSA may be efficient, they may not scale up to large datasets or may be incomplete.

6 Conclusions and Future Work

We have studied the skyline problem in large datasets and proposed a ranking algorithm that provides an efficient solution to this problem. Empirically we studied the performance and quality of our solution on synthetic data. Our experimental results suggest that our proposed algorithm overcomes existing approaches by up to two orders of magnitude while the skyline is completely generated. In the future we plan to exploit properties of variations of R-trees and trie-based indexes, like SkyMap [15], to improve the performance of our approach and apply the proposed techniques in different real-world domains. Additionally, we plan to equip the FOPA's approach with dynamic structures and thus, enable the algorithm to work with streaming data in dynamic environments.

References

1. Abadi, D.J., Boncz, P.A., Harizopoulos, S.: Column oriented database systems. PVLDB 2(2), 1664–1665 (2009)
2. Balke, W.-T., Guntzer, U.: Multi-objective query processing for database systems. In: VLDB, pp. 936–947 (2004)
3. Balke, W.-T., Güntzer, U., Zheng, J.X.: Efficient distributed skylining for web information systems. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 256–273. Springer, Heidelberg (2004)
4. Bentley, J., Kung, H., Schkolnick, M., Thompson, C.: On the average number of maxima in a set of vectors and applications. Journal of ACM (JACM) (1978)
5. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering, pp. 421–430. IEEE Computer Society, Washington, DC (2001)
6. Chen, L., Gao, S., Anyanwu, K.: Efficiently evaluating skyline queries on RDF databases. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 123–138. Springer, Heidelberg (2011)
7. Chen, L., Lian, X.: Dynamic skyline queries in metric spaces. In: EDBT, pp. 333–343 (2008)
8. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE, pp. 717–719 (2003)
9. Fuhry, D., Jin, R., Zhang, D.: Efficient skyline computation in metric space. In: EDBT, pp. 1042–1051 (2009)
10. Godfrey, P., Shipley, R., Gryz, J.: Algorithms and analyses for maximal vector computation. VLDB J. 16(1), 5–28 (2007)
11. Goncalves, M., Vidal, M.-E.: Reaching the top of the skyline: An efficient indexed algorithm for top-k skyline queries. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 471–485. Springer, Heidelberg (2009)
12. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: International Conference on Very Large Data Bases (VLDB), pp. 275–286 (2002)
13. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting Stars: The k Most Representative Skyline Operator. In: ICDE, pp. 86–95 (2007)
14. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1), 41–82 (2005)
15. Selke, J., Balke, W.-T.: SkyMap: A trie-based index structure for high-performance skyline query processing. In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011, Part II. LNCS, vol. 6861, pp. 350–365. Springer, Heidelberg (2011)
16. Sheng, C., Tao, Y.: Worst-case i/o-efficient skyline algorithms. ACM Trans. Database Syst. 37(4), 26 (2012)
17. Skopal, T., Lokoc, J.: Answering metric skyline queries by pm-tree. In: DATESO, pp. 22–37 (2010)
18. Tan, K., Eng, P., Ooi, B.: Efficient progressive skyline computation. In: VLDB 2001: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 301–310. Morgan Kaufmann Publishers Inc., San Francisco (2001)
19. Vlachou, A., Vazirgiannis, M.: Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships. Data Knowl. Eng. 69(9), 943–964 (2010)

UMAP: A Universal Layer for Schema Mapping Languages*

Florin Chertes and Ingo Feinerer

Technische Universität Wien, Vienna, Austria
Institut für Informationssysteme
FlorinChertes@acm.org, Ingo.Feinerer@tuwien.ac.at

Abstract. Schema mappings are fundamental notions in data exchange and integration for relating source and target schemas. Visual mapping languages provide graphical means to visually describe such transformations. There is a plethora of tools and languages available however all use different notions and visualizations and are hardly extensible.

In this paper we propose a new universal layer UMAP for schema mapping languages which provides a unified abstraction and middleware for high-level visual mapping languages. We use only standardized UML and OCL artifacts which allow for easy code generation in a number of target languages (e.g. C++ code) and for a modular extension mechanism to support complex schema mappings. We illustrate our layer by translating key elements of CLIP, a recent expressive visual mapping language, and show that UMAP has enough expressive power to encode all CLIP features. Moreover, we outline a strategy for automating the translation of any visual input language with a formal meta-model to UMAP.

1 Introduction

Schema mappings are central notions both in data exchange and data integration. They provide a precise formalism for modeling and describing the process of transforming source instances to target instances of a database in an information exchange scenario. The most common formalism for expressing schema mappings are logical formulae, typically in first-order or second-order logic [1]. The use of logics allows for exact definitions of the syntax and semantics of schema mappings contributing to the success of data exchange in theoretical research during the last decade [2]. Similarly, schema mappings have been of tremendous importance in industrial data exchange applications, e.g., in the well-known IBM Clio mapping tool [3,4]. However, in an industrial context visual languages for modeling schema mappings have gained increasing importance over the last years. Visual languages hide logical formalisms behind graphical notations and allow users without deep technical and mathematical background to perform data exchange. This is especially relevant for big data applications as manual compilation and inspection becomes inherently complex with increasing

* This work was supported by the Austrian Science Fund (FWF), project P25207-N23.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 349–363, 2013.

schema and data size. One of the most influential approaches along this line is Clip [5], a visual language for explicit schema mappings. Clip defines a set of custom language elements modeling source-to-target and hierarchical schema mappings. Clip and similar visual language formalism have their merits for high-level modeling by providing appropriate visual elements for the most common mappings. Nonetheless, we observe a number of drawbacks. First, there is no unified formalism nor standard for the actual elements of such a visual mapping language: supported elements depend on the concrete schema mappings supported by the tool, and each visual language uses different visualizations for its elements. Second, when automatically generating code from schema mappings, various tools (Ibm Clio, Altova MapForce, Stylus Studio, etc.) differ significantly in the number of target languages and the concrete implementation of the rules. Finally, there is a lack of easy extension mechanisms that allow the user to model additional types of schema mappings, e.g., for second-order dependencies, or mappings in the non-relational case. Consequently, these challenging tasks need to be addressed to foster the applicability of visual languages for schema mapping design in industry. To the best of our knowledge no comprehensive middleware for visual schema mapping languages exists to fill this gap. Slightly related is [6] where schema mappings are deduced from user examples. We propose a new unifying layer for visual schema mapping languages based on standardized Uml class diagrams [7] and Ocl constraints [8]. This layer is intended as a middleware underlying high-level visual languages like Clip or schema mapping toolkits like Clio but can also be used directly to visually design, model, and maintain schema mappings. By using only standardized and well-understood artifacts (basic features of Uml class diagrams and selected Ocl constraints) from the Uml modeling language we obtain a precise syntax and semantics for our layer which can be translated back to logics [9,10]. Most existing Uml toolkits support the generation of code from class diagrams which we use for implementing our schema mappings in various target languages. Finally, our approach is modular and allows easy extension of new schema mappings and targets for code generation. Our main contributions are:

- The syntax and semantics of the Umap layer. We show how to model central elements occurring in common visual mapping languages via Umap following a generic strategy defined by the Umap semantics. As a recent and prominent example we use Clip as example input language: we map the core Clip language elements to our Uml-based formalism, demonstrating the translation of source-to-target mappings to Uml class diagrams augmented with Ocl-constraints.
- The handling of more complex transformations like joins with grouping in the context of nested schema mappings for tree-like data structures (e.g., necessary for Xml data sources) in our proposed formalism. These transformations are characterized by more involved restructuring operations to map the source schema to the target schema. We show that Umap has enough expressive power to capture all features of Clip.

2 Preliminaries

Schema Mappings and Dependencies. A *schema* $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a set of relation symbols R_i each of a fixed arity. An *instance* I over a schema $\mathbf{R}$ consists of a relation for each relation symbol in $\mathbf{R}$, s.t. both have the same arity. We only consider finite instances. Let $\mathbf{S} = \{S_1, \ldots, S_n\}$ and $\mathbf{T} = \{T_1, \ldots, T_m\}$ be schemas with no relation symbols in common. A *schema mapping* [11] is given by a triple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ where $\mathbf{S}$ is the source schema, $\mathbf{T}$ is the target schema, and Σ is a set of logical formulae called dependencies expressing the relationship between $\mathbf{S}$ and $\mathbf{T}$. Instances over $\mathbf{S}$ (resp. $\mathbf{T}$) are called *source* (resp. *target*) *instances*. If I is a source instance and J a target instance, then $\langle I, J\rangle$ is an instance of the schema $\langle \mathbf{S}, \mathbf{T}\rangle$. Given a (ground) source instance I, a target instance J is called a *solution for* I *under* $\mathcal{M}$ if $\langle I, J\rangle \models \Sigma$. *Source-to-target tuple generating dependencies (s-t tgd)* are logical formulae of the form $\forall \boldsymbol{x}(\phi(\boldsymbol{x}) \rightarrow \exists \boldsymbol{y}\psi(\boldsymbol{x}, \boldsymbol{y}))$.

Clip. Clip is a mapping language for relational and Xml schemas. Schema elements are visually connected from source to target by lines interpreted as mappings. Both structural mappings and simple value mappings are supported. The combination of value and structural mappings in Clip yields expressive language elements extending those from Clio [3,4], one of the most prominent schema mapping tools developed by Ibm Almaden Research Center and the University of Toronto, and gives users fine-grained control over generated transformations. Mappings are compiled into queries that transform the source instances into target instances. The main Clip language elements [5, Fig. 2] are as follows.

- *Value nodes* store attributes and text, like *value: String* or *@pid: int* in Fig. 1.
- *Single elements* consist of a *value node* and have a name. Examples are *pname*, *ename* or *sal* in Fig. 1.
- *Multiple elements* represent sets of elements. E.g. *Proj[0..*]* from Fig. 1.
- *Value mappings* are thin arrows with open ends, in order to map values from source to target. E.g. in Fig. 1 the arrow connecting *ename* with *@name*.
- *Builders or object mappings* are thick arrows with closed ends connecting elements. Examples are the two bold arrows in Fig. 1.
- *Build nodes* have at least one incoming *builder* and at most one outgoing *builder* and express a filtering condition in terms of the variables in the *builders* or enforce a hierarchy of *builders* if connected by *context arcs*.
- *Grouping nodes* are a special kind of *build nodes* used for grouping on attributes. Fig. 3b gives an example.
- *Context propagation trees* are trees with *build nodes* and *context arcs*. E.g. the arrow connecting the two *build nodes* in Fig. 3a.

3 UMAP Layer and Translation of Clip Core Features

Before defining any syntax or semantics the main high-level idea behind Umap is summarized as follows. The Umap layer abstracts source and target schemas as

Uml class diagrams. W.l.o.g. we assume both source and target as Xml schemas (since relational schemas can be converted into Xml schemas). The schemas represent trees consisting of nodes, attributes and sets of nodes. Individual schema elements, i.e. nodes, are modeled as classes, and attributes in the Xml schema become attributes in the corresponding Uml class. Sets of elements are modeled as generic container classes encapsulating the underlying class. The actual mappings between source and target schemas are done by association classes augmented by associations between them. We use Ocl to specify constraints (post conditions and invariants) on the association class functions and class attributes to achieve the desired semantics of the mapping.

We will start with an example to translate basic features of Clip into Umap. The exact definition of the Umap language follows in the next section once we have motivated a set of typical constructs needed; Clip is a good representative for recent visual mapping languages and has clearly motivated the need for the individual language constructs. We map each Clip artifact to a Uml/Ocl artifact of the Umap layer. The Clip *value nodes* and *single elements* are modeled by class attributes grouped semantically in a class. The Clip *multiple elements* are modeled in Uml as generic container classes (sets of elements). The *value mappings* are modeled in Uml with the help of an association class linking source to target. A class function named *map* implements the mapping. The definition of the mappings is achieved through Ocl expressions which include also the filtering conditions. The *builders* or *object mappings* are modeled in Uml also with the help of an association class linking source to target. A class function named *build* implements the iterator defined by Ocl expressions. The associations between association classes model the hierarchy of *builders*. The *context propagation tree* is achieved with the help of the hierarchies of association classes and associations between them. Each iterator modeled by a class function named *build* from one level of the hierarchy triggers only a class function named *map* from a lower level in this hierarchy which maps source to target values. The class function named *map* from that level triggers zero or more class functions named *build* from a lower level of the induced hierarchy of functions. As a general characteristic of the translation from Clip to Uml the translations of the Clip *value mappings* and *builders* are associations classes using functions named *map* or *build*. Successive alternations of these two functions correspond to the Clip feature of a *context propagation tree*. Joins and *grouping nodes* are modeled with the help of the Ocl expressions defining class functions and attributes.

A Simple Clip Mapping. A simple mapping is presented in Fig. 1 (adapted from [5, Fig. 3]): an *employee* is created for each *regEmp* whose salary is greater than *11000*. For each employee the name is also copied from source to target. In [5] is explained that this mapping is expressible in Clio. The authors mention further that this mapping is underspecified: there is no indication how to map the *dept* from the source to *department* in the target. Using the notion of *universal solution* [1] the authors explain that there are at least two such solutions: an universal solution with a generic *department* for each mapped *employee* or an universal solution with a single generic *department* for all mapped *employees*.

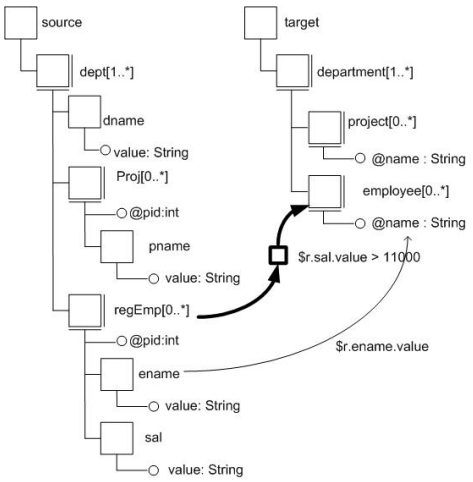


Fig. 1. A simple Clip mapping (adapted from [5, Fig. 3])

By adopting the principle of *minimum-cardinality*, the authors prefer the latter solution. For clarity we present the mapping of the source set *regEmp[0..*]* to the target set *employee[0..*]* without the source *dept* and the target *department* to which they belong. Thus we represent only the essential part of Fig. 3 from [5] using a class diagram in Fig. 2.

The Motivating Example: A Simple Mapping. The Uml class diagram in Fig. 2 presents the structure and the Ocl expressions define the operations used to map the source to the target. There are two classes on the source side: a class of type *regEmpSet* and a class of type *regEmp* connected with the previous by aggregation with cardinality *0..**. The class *regEmp* contains two attributes: *ename* of type *string* and *sal* of type *int*. On the target side there are two classes: *employeeSet* and *employee* connected by aggregation with the same cardinality as the previous aggregation from the source side. The class *regEmpSet* from the source is connected to the class *employeeSet* from the target by an *association class*: *Builder*. In the same way the class *regEmp* from the source is connected to the class *employee* from the target by an *association class*: *ValueMap*. Between these two association classes there is an *association* which helps the class *Builder* to access the functionality of the class *ValueMap*. The *Builder* association class iterates through the source set using the function *build*. In each iteration by the help of the association class *ValueMap* each *regEmp* is mapped to an *employee* using the function *map*. These both functions, *Builder::build* and *ValueMap::map* are defined by Ocl post-condition expressions.

Translating *Value Nodes*, *Single Elements* and *Multiple Elements*. The previously named classes translate the Clip structure to Uml. Both *Set* classes: *regEmpSet* and *employeeSet*, represent the *multiple nodes* in Clip language: *regEmp[0..*]* and *employee [0..*]*. The other two classes *regEmp* and *employee*

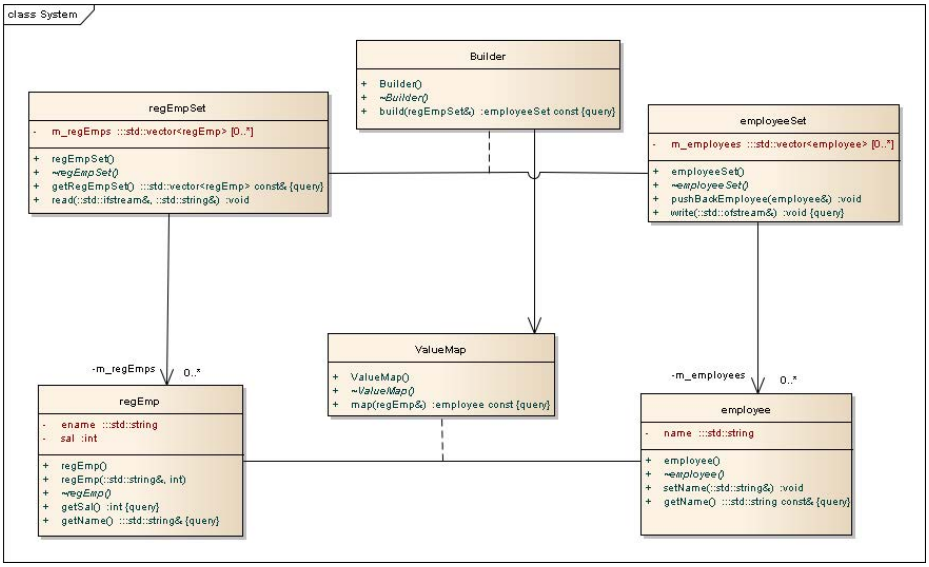


Fig. 2. The class diagram with a simple mapping corresponding to Fig. 1

put together all *value nodes* and *single elements* that structurally belong to the *multiple elements* such as *regEmp[0..*]* and *employee [0..*]*.

Translating *Value Mappings* and *Builders*. The semantics of Clip *value mappings* and *builders* is achieved in the Uml translation through artifacts of the class diagram and the Ocl expressions. We use in the class diagram the association class *ValueMap* that connects the source type *regEmp* to the target type *employee*. In the Uml translation of Clip *value mapping*, the generation of the target object from the source object is done by the help of the function *ValueMap::map*. This mapping function is defined in Ocl as follows:

```
context ValueMap::map(rEmp: regEmp): employee
post: result = e: employee and e.name = rEmp.ename
```

In Ocl [8, Section 7.6.2] it is mentioned that an operation could be defined by a postcondition. The object that is returned by the operation can be referred to by the keyword `result`. In our case the target to source mapping is defined by the equality of the names. Other mapped attributes could be here similarly detailed if necessary.

In the Uml translation of Clip *builder*, the generation of the target set from the source set is achieved by the use of the function *Builder::build*. This function iterates over the set *regEmpSet* generating the set *employeeSet* and by this models the Clip *builder*. This mapping function is defined in Ocl as follows:

```
context Builder::build(rEmpSet: regEmpSet): employeeSet
post: result = rEmpSet.m_regEmps->select(r | r.getSal() > 11000)
  ->collect(r: regEmp | ValueMap.map(r): employee)
```

The mapping definition starts from the source set *regEmpSet* and selects only those objects from the source that have a salary greater as 11000 creating a set. In the next step we obtain another set of type *employeeSet* from this set. This is done by the use of the function *collect* that applies to each object of type *regEmp* the function *ValueMap::map*. The result is an object of type *employee*. Further the function *collect* inserts all this newly created objects into a set which is the return value of the function *Builder::build*. The class *Builder* is the translation of the CLIP *builder* because it iterates on the source set, it selects the nodes to be mapped to the target by the help of the OCL *select* function and then it creates a totally different set using the OCL *collect* function. The function *collect* uses the association to the class *ValueMap* to effectively map each object from the source to the target. The class *ValueMap* translates the CLIP *value mappings*. The presentation of this example used implicitly the modeling language for schema mapping UMAP which is described in the following.

4 The UMAP Language

The previous section showed how to reproduce some central building blocks of a visual modeling language just with UML and OCL constructs. However, in order to provide a well-defined foundation for a middleware like UMAP, we need to restrict the allowed UML constructs and provide a clear generic strategy of translating the input language constructs to UML and OCL expression generating the same results: syntax and semantics.

The Language Syntax. The UMAP language uses a small and well-defined subset of UML and OCL constructs. The main UML building blocks:

- *class*,
- *association class*,
- *association* and a special form of it called *aggregation*,

as defined by the official abstract syntax of the language [7, Section 7] are used by the UMAP language for describing the structure of the source and the target, and for the mapping from source to target. Moreover, the OCL constructs *asSet*, *collect*, *forAll*, *isUnique*, *iterate*, *result*, and *select*, are part of UMAP. According to UML [7, Section 2.1] such a visual language consists of *language units* which are adapted to a particular paradigm or formalism. Between these language units there are no interdependencies, so they can be used one apart from the other. Some of these language units are partitioned into multiple increments conducting to a horizontal stratification of UML. These layers of increasing capability are called *compliance levels*. In UML [7] there are four compliance levels. UMAP uses the language unit *Classes*, and adheres to the fourth level of compliance named *L3* because a main construct of UMAP is the *association class* from the meta model package *Classes::AssociationClasses*.

The *classes* of UMAP describe the structure of the source and the target. In UMAP the aggregation (represented as an association without the diamond

notation) is the only connection inside the source or target structures. That means that the structure at the top level includes arrays of other structures that again include arrays. This description defines the source and the target structures as trees. The behavior of the *association class* is used for the actual mapping. The only active class is the *association class* at the top of such a hierarchy. We use the property described in [8, Section 7.6.3] that from an *association class* we can navigate the association-ends. This comes in contradiction with [7, Section 7.3.4] that states the contrary. No matter the way this contradiction between these two standards will be solved in the future by the standard committees the syntax and semantics of UMAP is not affected and minor adapting changes are necessary only if [8, Section 7.6.3] is going to be modified and aligned with [7, Section 7.3.4].

There are two different usages of the *association class*: one with a function named *build* and the other named *map*, building two types. The first type connects top level structures from target to source. At the same time this first type is connected through an *association* only to one object of the second type from a lower level in the tree hierarchy. The second type is connected in the same way to zero or more objects of the first type also from a lower level of the tree hierarchy.

The Language Semantics. The semantics of the UMAP language describes the transformation of the source into the target structure. As we claim standard compliance with UML its semantics uses the standard behavior [12, Section 7.11]. Using the definition of *attribute grammars* [13], additional semantics of UMAP was achieved through constraints expressed in the OCL language for the following constructs of the UMAP language:

- behavior constructs i.e. the functions named *build* and *map* of the construct *association class* and
- structure constructs i.e. attributes of the target structure.

The OCL expressions are implemented in a programming language that can be executed to create and instantiate a mapping. The usage of OCL in UMAP is limited to following basic constructs: *asSet*, *collect*, *forAll*, *isUnique*, *iterate*, *result*, and *select*. For their semantics we assume the default interpretation as defined by the OCL standard [8].

Proposition 1. *Another characterization of the semantics of the* OCL *constructs in* UMAP *is given by logics.*

Proof (Sketch). The translation of the OCL constructs *asSet*, *collect*, *forAll*, *isUnique* and *select* to first-order predicate logic is given in [10]. The construct *result* is used only for defining the output. The authors in [10] do not translate the operator *iterate* to first-order predicate logic but propose to express it in higher-order logic.

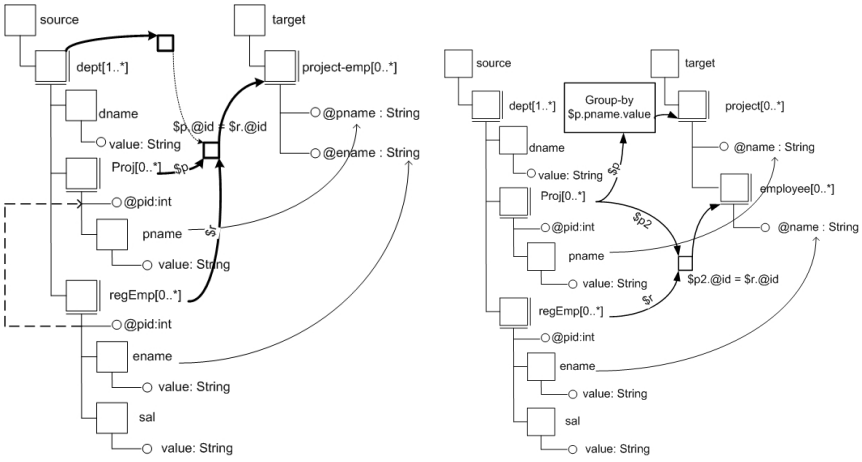
The active class is the *association class* connecting the top level structures from the source to the target. This *association class* must be from the first type so it must have a function named *build* and it is an iterator on included arrays. This

function is described in OCL and makes a selection of the source objects and a transformation of them by calling the other type of *association class* which is connected by an *association*. On its turn the second type of the *association class* using the function named *map* does the mapping. This is again described in OCL. A second role of the function *map* is to access through the *association* other objects of the *association class* of the first type having a function named *build*. This chain of these two types calling each other executes the mapping, thus defining the semantics of the language UMAP. A program in UMAP is a UML diagram with OCL defining the mapping functions. An input is an instance of the source structure or an instance including such an object. The output of the program is an instance of the target structure. The semantics of the language is the mechanism that transforms the source into the target.

The Language Complexity. As the mappings also use OCL expressions the main consideration is their complexity when implementing them in a programming language. However, as we restrict the allowed constructs (as defined above) the query complexity of the mapping is polynomial. In fact, the implementation of the OCL functions *iterate*, the most general, or other more specialized as *select* and *collect*, can be done by a bounded number of nested loops.

5 Complex Mappings and Tree-Like Data Structures

In contrast to the previous CLIP features, the next mappings join multiple source elements and create multiple target elements. This means that both functions *map* and *build* create sets of the same type, with the consequence that both *association classes* containing these functions are connected on the target side



(a) A CLIP mapping with a join constrained by a CPT

(b) A CLIP mapping with a join and grouping

Fig. 3. Two mappings with joins (adapted from [5, Fig. 6]) and [5, Fig. 7])

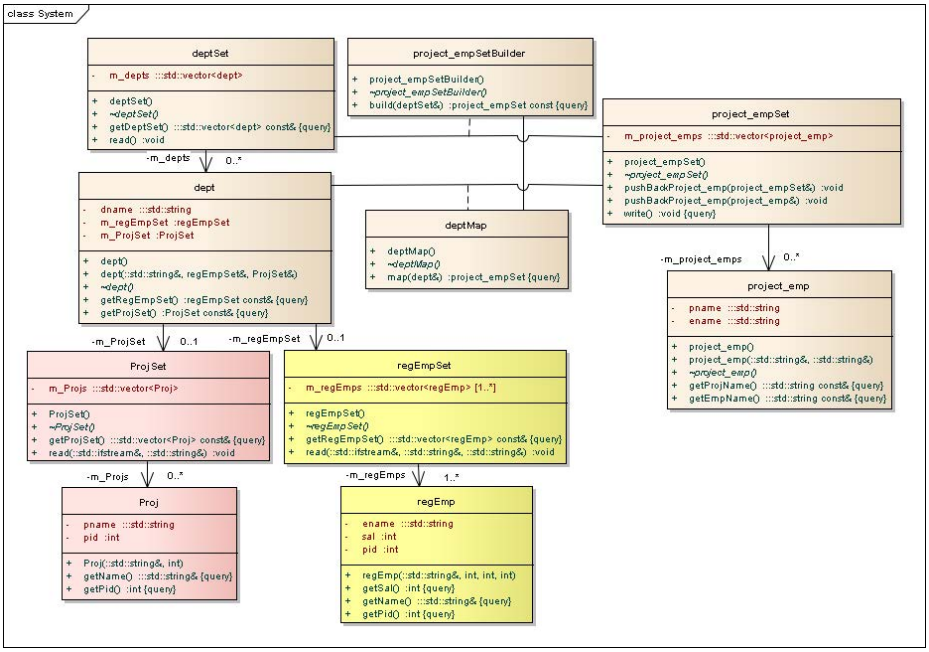


Fig. 4. The UMAP diagram for a join

to the same class. The function *map* uses the join to map from target to source, while the function *build* iterates over the set *deptSet* and inserts target sets produced by the function *map* into the union of target sets.

Before discussing these more complex features for illustration in detail, we show a general result that the expressive power of UMAP captures all language features of CLIP.

Theorem 1. *Let* $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st} \cup \Sigma_t)$ *be a schema mapping where* Σ_{st} *denotes a set of s-t tgds and* Σ_t *is a set of target tgds. Let* I *be a source instance over* $\mathbf{S}$ *and let* J *be a solution over* $\mathbf{T}$ *satisfying* $\mathcal{M}$*, i.e.,* $\langle I, J\rangle \models \Sigma_{st} \cup \Sigma_t$*, obtained via the execution of* CLIP*. Then there exists a solution* J' *over* $\mathbf{T}$ *with* $\langle I, J'\rangle \models \Sigma_{st} \cup \Sigma_t$ *obtained via the execution of the corresponding* UMAP *model such that there is a homomorphism from* J' *to* J*.*

Proof (Sketch)

– A *simple* CLIP *mapping* [14, Fig. 1] is by construction equivalent with the UMAP translation [14, Fig. 2]. The *value mapping* and the function *map* have the same functionality, by mapping values to values. The *builders* and the *build nodes* construction define an iterator identically as the UMAP function *build*. In this case the s-t tgds, Σ_{st}, require that all elements of the source satisfying a certain condition are to be mapped to the target. The exact mapping is also defined. Both CLIP and UMAP implement the same s-t tgds.

- Clip mapping with one *context propagation* [14, Fig. 3] is by construction equivalent with the Umap solution [14, Fig. 4]. The basic concept of Umap uses the *associations* to connect the *associations classes* from one level of the model hierarchy to the next level by this simulating the *context propagation*.
- Clip mapping with multiple context propagations [14, Fig. 5] is by construction equivalent with Umap [14, Fig. 6]. The previous concept can be easily extended when we change from one *context propagation* to many *context propagations* building a hierarchy.
- Clip mapping with a join constrain by a CPT [14, Fig. 7] and its Umap translation [14, Fig. 8] are defined by the same s-t tgds. In this case the Clip join is by definition and construction identical to the Umap defined join. We give more details on this case in the discussion of this section.
- Clip mapping with a grouping and join [14, Fig. 9] and the Umap translation [14, Fig. 10] define the s-t tgds differently. The Clip mapping uses a *Skolem function* and the Umap mapping uses supplementary to the s-t tgds target conditions. In Umap the mapping function is defined by the conditions imposed on the resulting target instance. A detailed description of this case is given below, so that we can always find a homomorphism between the solutions obtained from Clip and Umap.
- Clip mapping with inverting the nesting hierarchy [14, Fig. 11] and the Umap translation [14, Fig. 12] use the same technique as in the previous case, Clip uses a *Skolem function* and the Umap mapping functionality is deduced by the s-t tgds and target conditions.
- Clip mapping with aggregates [14, Fig. 13] and the Umap translation [14, Fig. 14] use again different technics obtaining the same target instance. Clip uses *Skolem functions* and in Umap creates the mapping using the definitions of the aggregate functions.

A Join Constrained by a CPT. A join in Clip is achieved by a *context propagation tree* (CPT) as shown in Fig. 3a. The result is a flattened list of employees and projects in which they work. The Umap class diagram in Fig. 4 has on the target side the class *project_empSet*. As in Clip, *dept* is not mapped in the target. When a *build node* is reached from two or more *builders*, Clip computes a Cartesian product or a join if a condition involving two different variables is present. The Uml translation is based on the definition of the Cartesian Product in Ocl by [15]. The association class *project_empSetBuilder* starts the iteration over the elements of the set *deptSet*. Each iteration maps one object of type *dept* to a set of objects *project_emp* obtained from the join of the *Proj* and *regEmp* objects of each *dept* on the attribute *pid*. The Ocl expressions define the join by constructing first a Cartesian Product and then a selection of the elements with the same attribute *pid* associating each employee with the projects in which he works. In this case the association class *deptMap* creates from each object *dept* a set of *project_emp*. The association class *project_empSetBuild* using this functionality maps the set of *dept* objects to the union of sets of *proj_emp* objects. We define the Ocl expression for the function *deptMap::map* as

```
context deptMap::map(dep: dept): project_empSet
def: projProdEmp = dept.m_ProjSet->collect(p: Proj | dept.m_regEmpSet
  ->collect(e: regEmp | Proj_regEmp: Tuple {Proj, regEmp}))
```

This is the Cartesian Product of the two sets included in an *dept*. The result is a set of tuples composed of a *regEmp* and a *Proj* each.

```
def: projJoinPidEmp = projProdEmp
  ->select(Proj_regEmp | Proj_regEmp.Proj.pid = Proj_regEmp.regEmp.pid)
```

The join is obtained by its definition from the Cartesian Product by selecting those tuples with the same *pid*.

```
post: result = projJoinPidEmp->collect(Proj_regEmp|project_emp(
  Tuple {pname = Proj_regEmp.Proj.pname, ename = Proj_regEmp.regEmp.ename}))
```

The result of this operation is a set of *project_emp* objects containing the name of the project and the name of the employee working in that project. The OCL definition of the function *project_empSetBuilder::build* uses the function *deptMap::map*.

```
context project_empSetBuilder::build(dSet: deptSet): project_empSet
post: result = dSet.m_depts->Set()->iterate(r: dept;
      peS: project_empSet = {} | peS.pushBackProject_emp(deptMap.map(r)))
```

This function iterates over the set of *dept* objects and produces from each of them a set of *project_emp* objects and these sets are inserted in the *project_empSet*, a union of sets. This ensures that the CLIP join and the described UML class diagram produce the same mapping.

A Mapping with Grouping and Join. *Group nodes* are used to group source data on attributes. Fig. 3b depicts such a construct. The result of a *group node* is a sequence of elements selected by the grouping attributes. The number of created sequences on the target equals the number of distinct values of the grouping attributes from the source. In Fig. 3b the *Projs* are grouped by *pname*. The *Projs* and *regEmps* are joined by *pid* and finally the *employees* on the target are created and added to the *project* by name independently of the *dept* in which they work. The class diagram in Fig. 5 is the corresponding translation for grouping in CLIP. The association class *projectSetBuilder* starts the iteration over the elements of the set *deptSet*. Each iteration using the function *deptMap::map* maps one object of type *dept* to an object of type *projectSet*. This set is obtained from the join of the *Proj* and *regEmp* objects of each *dept* on the attribute *pid*. On the target each *project* includes its *employee*. The function *deptMap::map* inserts each *project* into the *projectSet*. Each insert groups the *project* objects by name. In this case OCL expressions do not give a constructive solution but the OCL constraints define the possible implementations. The attribute *m_projects* of the type *projectSet* from the target is specified in OCL by the following expression:

```
context projectSet.m_projects inv: self->isUnique(p: project | p.name )
```

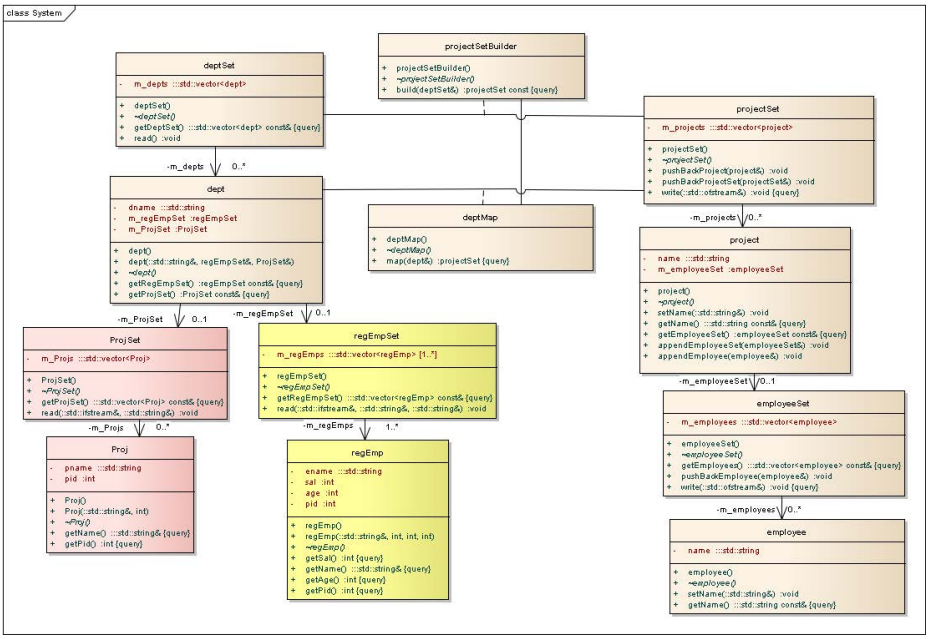


Fig. 5. The UMAP diagram for a join and grouping

This means that the elements of the set, the *project* objects, are unique by name. In this way the grouping by project name is achieved. In the UML diagram the type *project* has a set of objects of type *employee*. Because of this structure the only possible grouping is to attach all the employees to the project in which they work. If two or more projects have the same name by the uniqueness of the project name the employees of this projects are again grouped together. This is valid by the structure of the UML diagram also for projects in different departments. The OCL expressions give the definition of the join by constructing first a Cartesian Product and then a selection of the elements with the same attribute *pid* associating each employee with the projects in which he works. It follows the OCL expression for the function *deptMap::map*:

```
context deptMap::map(dep: dept): projectSet
```

The OCL expressions defining the Cartesian Product and the join on *pid* have already been presented in the previous subsection.

```
def: result_lhs = projJoinPidEmp->collect(Proj_regEmp |
  Tuple {pname = Proj_regEmp.Proj.pname, ename = Proj_regEmp.regEmp.ename})
```

This OCL expression creates all the tuples from the source that are to be grouped on *project* name in the target by the mapping,

```
def: result_rhs = projectSet.m_projects->collect(p | p.m_employeeSet.m_employees
 ->collect(e | proj_emp: Tuple {p: project, e: employee}))
```

creates the Cartesian Product of the tuples from the target, and

```
post: result = projectSet(result_lhs) and
result_lhs->forAll(pe|result_rhs->exists(proj_emp |
 proj_emp.pname = pe.pname and proj_emp.ename = pe.ename))
```

defines the constraint that all tuples from the source must have a correspondent in the target. All elements from the target are created only from the source so it is not necessary to show that all elements from the target are only those that are created by the mapping. Every possible implementation must fulfill these constraints. The association class *deptMap* connects the class *dept* from the source with class *projectSet* from the source. The function *deptMap::map* transforms a *dept* to a *projectSet*. The association class *projectSetBuilder* connects the class *deptSet* from the source with class *projectSet* from the target.

```
context projectSetBuilder::build(dSet: deptSet): projectSet
post: result = dSet.m_depts->Set()->iterate(r: dept;
      pS: projectSet = {} | pS.pushBackProjectSet(deptMap.map(r)))
```

The function *projectSetBuilder::build* transforms the source to the target. The OCL definition of the function *projectSetBuilder::build* uses the function *deptMap::map*.

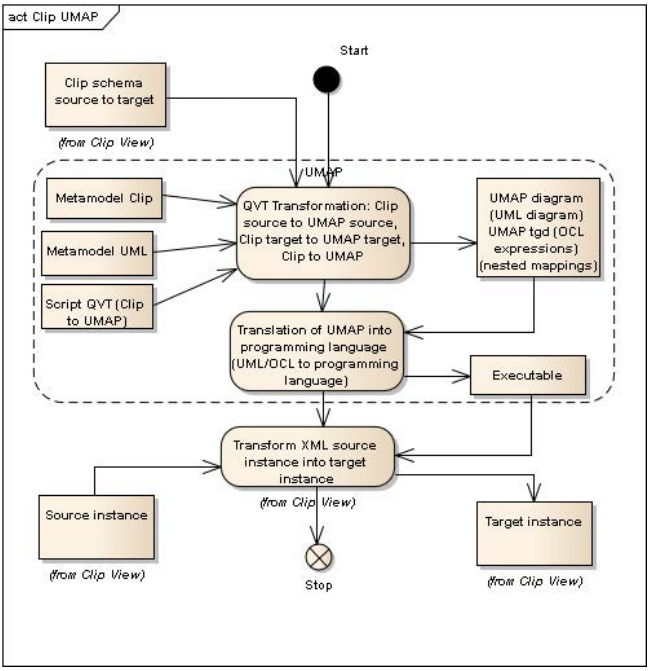


Fig. 6. UMAP transformation for CLIP

6 Conclusion

In this paper we have introduced UMAP, a new universal layer for schema mapping languages. Schema mappings are modeled by standardized UML class diagrams and OCL expressions. By restricting the UML artifacts to well-understood elements (e.g., classes, associations, aggregations, class functions, and straightforward post-conditions and invariants), there is a well-defined semantics. This allows us to translate UMAP specifications to a broad range of target languages (like C++). We have modeled a set of common schema mapping operations in UMAP, like basic source-to-target dependencies, join, and grouping operations. We translated several core features of CLIP to UMAP (see [14] for a translation of all CLIP artifacts). There is also an implementation available (see `http://www.dbai.tuwien.ac.at/research/project/umap`) generating C++ code showing the translation of typical CLIP language elements to our UML-based formalism, proving that our approach works in practical usage. UMAP can be seen as a new middleware for high-level visual schema mapping languages. We propose to use UMAP as a back-end when creating a new visual mapping language, as with a formal meta-model it can be rather easily automatically mapped to UMAP via QVT, an evolving standard for Query/View/Transformation. The high-level process for using QVT with UMAP is depicted in Fig. 6.

References

1. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing schema mappings: Second-order dependencies to the rescue. ACM TODS 30(4), 994–1055 (2005)
2. Lenzerini, M.: Data integration: a theoretical perspective. In: PODS. ACM (2002)
3. Popa, L., Velegrakis, Y., Hernández, M.A., Miller, R.J., Fagin, R.: Translating web data. In: CAiSE 2002 and VLDB 2002, pp. 598–609. Morgan Kaufmann (2002)
4. Fuxman, A., Hernandez, M.A., Ho, H., Miller, R.J., Papotti, P., Popa, L.: Nested mappings: schema mapping reloaded. In: VLDB, pp. 67–78 (2006)
5. Raffio, A., Braga, D., Ceri, S., Papotti, P., Hernández, M.A.: Clip: a visual language for explicit schema mappings. In: ICDE 2008, pp. 30–39 (2008)
6. Alexe, B., ten Cate, B., Kolaitis, P.G., Tan, W.C.: Designing and refining schema mappings via data examples. In: SIGMOD Conference, pp. 133–144 (2011)
7. OMG: Unified Modeling Language Superstructure 2.4.1. (2011), `www.omg.org`
8. OMG: Object Constraint Language 2.3.1. (2012), `http://www.omg.org`
9. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)
10. Beckert, B., Keller, U., Schmitt, P.: Translating the Object Constraint Language into first-order predicate logic. In: VERIFY, FLoC (2002)
11. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. 336(1), 89–124 (2005)
12. OMG: Unified Modeling Language Infrastructure 2.4.1. (2011), `www.omg.org`
13. Meyer, B.: Introduction to the Theory of Programming Languages. P.-H. (1990)
14. Chertes, F.: DBAI-TR-2012-76. Technical report, DBAI, TU Wien (2012)
15. Akehurst, D.H., Bordbar, B.: On querying UML data models with OCL. In: Gogolla, M., Kobryn, C. (eds.) UML 2001. LNCS, vol. 2185, pp. 91–103. Springer, Heidelberg (2001)

GCAPM: A Generic Context-Aware Model in Peer-to-Peer Environment

Saloua Zammali[1], Khedija Arour[2], and Amel Bouzeghoub[3]

[1] Faculty of Science of Tunis, University Tunis El-Manar, 2092 Tunis, Tunisia
zammalisalwa@gmail.com
[2] National Institute of Applied Science and Technology of Tunis, 1080 Tunis, Tunisia
Khedija.arour@issatm.rnu.tn
[3] Department of Computer Science, Télécom SudParis, UMR CNRS Samovar, 91011 Evry Cedex, France
amel.bouzeghoub@it-sudparis.eu

Abstract. Many different applications are concentrating on collecting information about users. The main purpose of this information collection is to enable the applications of understanding the users, their preferences and their interests in order to provide them with personalized services. For these reasons, different applications in several areas organize user properties, preferences and interests based on a user model, a structure holding all relevant user-related information. Thus, the issue of user modeling is being tackled by researchers attempting to propose valuable structure and to provide adaptable applications to use it successfully. However, the most of published studies are focused on centralized user model. In this sense, we propose a Generic Context-aware Model (GCAPM) in peer-to-peer (P2P) environment. We illustrate the use of the model in the context of a case study, focusing on query expansion.

Keywords: User model, Prediction, Ontology, Expansion, P2P Systems.

1 Introduction

In different fields of the Web, personalization and adaptation are crucial concepts nowadays because they help users to find what they really want and need. In many areas (digital libraries, search engines, e-learning, online databases, e-commerce, social networks, etc), users have to adapt techniques and visualization content to their own needs. A simple way to capture the user context (preferences, interests, location, etc) and differences in the needs of individual users is ensured by user modeling.

User modeling within pervasive environments has been previously studied within many research areas. Most of them are gathered in the survey done in [6]. However, these models have not been widely applied to the mobile application area.

In a peer-to-peer environment, contexts are distributed in respective terminals, which is different from traditional computing architecture with a central

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 364–371, 2013.

server. In peer-to-peer systems, there exist a small number of works which address the problem of context modeling. In [5], [2], [7], the authors propose a methods which consist of building user model based on user's interests. The main limitation of such approaches is that they don't take into account the user's contextual information. In the best of our knowledge, the only work that models the user in a mobile environment, taking into account the user's context is [9]. However, this work lacks of taking into account the specifity of P2P systems.

To solve these problems, this paper presents a generic P2P user modeling, called GCAPM (Generic Context-Aware Peer Model). In this paper, we address the problem of setting up a generic and extensible user model. Indeed, our model supports a variety of a context-aware applications (Aggregation, recommendation, etc) and it is possible to add new dimensions to the model.

The main contribution of the paper is to propose a user model based on the following requirements:

- The proposition of GCAPM, a generic user model.
- The construction of users interests centers.
- The instantiation of GCAPM model.
- The exploitation of the user model to queries expansion step for Peer-to-Peer Information Retrieval (P2PIR).

The rest of this paper is organized as follows. In section 2, we present our proposed model. In section 3, we discuss in more detail a use case in the context of query expansion. Finally, we conclude and give some directions for future work in Section 4.

2 GCAPM Model

From the related work, we can deduce that none of these user models fulfills the requirements needed for a generic user model. Only [9] adress many requirements and represent several contextual dimensions. However, proposed modeling in this case does not include all specific requirements in P2P systems.

Since modeling users, is a very complex task, an investment is needed to use some dimensions together. At the same time, we also aim at bridging the space left by the previous works by considering all dimensions that can be useful to perform any task in peer-to-peer information retrieval (recommendation, aggregation, filtering, expansion, etc).

To provide a user with a personalized P2PIR task, we need to take into consideration a wide range of the user's characteristis such as a contact information (including name, age, sexe, etc), profile, resource and context. Based on these characteristics a generic user model, *GCAPM*, has been proposed, as presented in Figure 1. A class diagram is used to clarify and represent relationships between user-related classes. We detail in the following, the CCAPM' components:

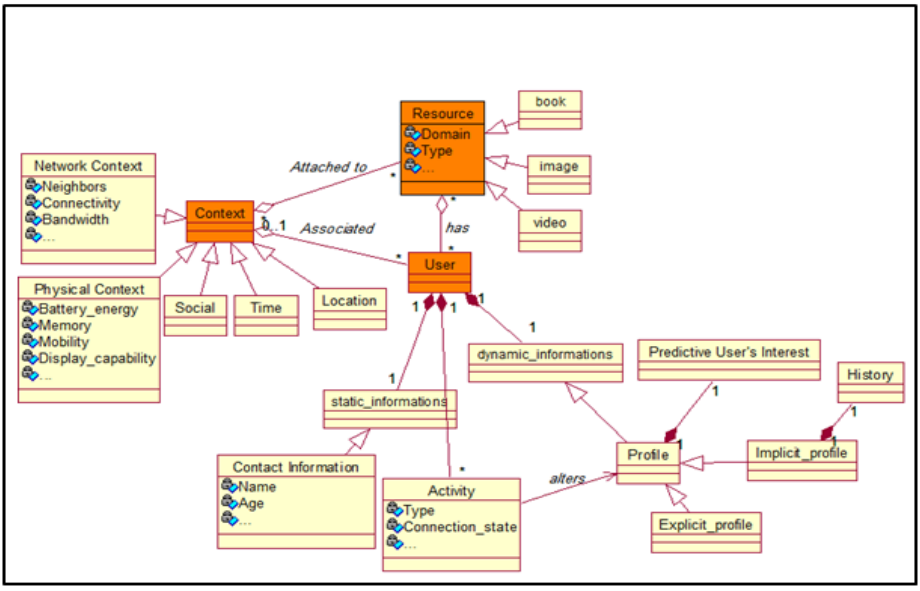


Fig. 1. GCAPM model diagram

– **User:** It is composed of
  - User's static information: refers to attributes used to personalize the search such as name, age, sex, etc.
  - User's dynamic informations: This component describes implicit and explicit profile. Implicit profile is based on user activity and history. User's activity is important and can influence his needs. It can be captured (*e.g.*, changes in location) and/or obtained using predective model. User's history provides details of users (*e.g.*, time of browsing sites, details of user accessed sites). When these needs are known in advance, a system can better adapt any result to its users. Based on the importance for related-activity information, we consider the activity and history are an important dimensions in modeling user.
    Explicit profile may useful in some cases (*e.g.*, cold start); therefore we consider, this dimension in user model.
  - Predictive User's Interest: Predictive user's interest is based on analyzing historical and current data and generates a model to help predict future outcomes. The knowledge of the user profile can help in predicting searches future intentions of the user to adjust the results according to these needs.
– **Context:** Contextual dimensions are considered a key issue with respect to the interaction between human and computer because they describe the surrounding facts or assumptions which provide a meaningful interpretation to a users' computer usage when their physical environment varies.

– **Resource:** Information retrieval and the related domains of information filtering are concerned with the finding of information, often in the form of *text* resources, which are in some sense about a topic that a user is interested in. In recent years, *image* collections available on the Internet, have increased both in number and size, and represent a huge amount of important information. The advancement of multimedia and enormous information present in the *video* requires video resource modeling. The context is described by: name, manufacturing environment (information related to the creation of the document), author, creation date, number of pages (in a book resource), editorial context (publication home), extension, dimension, duration (in video resource), domain, etc.

3 Case Study

GCAPM is a generic framework that supports a variety of context-aware applications (filtering, expansion, aggregation, recommendation, etc) in P2PIR systems.

a- User's interest : To address differences among users, there has been a recent surge of interest in personalized search to customize search results based on a user's interest.

b- User's history : Web search engines consistently collect information about users interaction with the system: they record the query they issued, the URL of presented and selected documents along with their ranking. This information is very valuable: Query logs have the potential to partially alleviate the search engines from thousand of searches by providing a way to predict answers for a subset of queries and users without knowing the content.

Before presenting the case, we give an example of GCAPM instantiation.

3.1 GCAPM Instantiation

In order to use the model in the field of P2PIR, in any step (like recommendation, expansion, filtering, etc), we need to instantiate it. For the instantiation of our model, we can use several information sources. In following, we detailled these sources.

User

a- Contact information We investigate publicly available data from social networks in order to provide an approximate substitute for real users. We aggregate information from multiple social networks to provide a test collection containing users contexts which could be used to evaluate P2PIR. Social networks do not support sharing of personal context due to privacy concerns. However, we could find public datasets containing data collected by public web pages from social networks. In order to abtain data covering the context dimensions listed in the section 2, we choose to use information from three data sources, BookCrossing, FriendFeed and users FOAF profile.

b- Activity Activity component is composed of two parts: activity type and connection state. We need a model for representation a user connection state. For example, OPO[1] (Online Presence Ontology) provides the main concepts and properties required to describe information about user's presence in the online world.

c- User Profile The user profile can be explicit (*i.e*, user's interests) and/or implicit (*i.e*, history). In the following, we explain how user's interests and profile are generated in the two cases (explicit and implicit).

Some approaches obtain user's interest by analysing URLs that user has browsed, like the work [3]. These profiles lack semantic information which makes it difficult to accurately represent user's interests. Other works used a questionnaire or filling a form. But, these informations are poor. We need to enrich these to have user's centers.

To built the centers of interest based on the documents judged relevant by the user, we adapted the mechanism proposed in [1].

For **implicit profile**, a history is information stored in a log file for future use or user habit analysis. To keep track of user interactions with peer-to-peer information retrieval system, we use a log file built in our team [8]. This file is formalized as follows: $\prec$ Query identifier, Query terms, Documents downloaded, Associates peer $\succ$. Upon receipt of a response on a given query, the module updates the log file by adding information to this query, *i.e.*, the identifier of the query, the theme (derived from whole its keywords), the documents downloaded by the user and associates peer.

Context. Modeling time is important in user modeling. We plan on using the *OWL-Time* ontology. OWL-Time[2] ontology ensures a good representation of the temporal information and its manipulation. It has become a reference for representing and reasoning about time.

To define basic dimensions and social aspects characterizing users, a clear model for representation and reasoning about them is necessary. We use *FOAF*[3] (Friend of A Friends) ontology to describe persons, their activities and their relations to other people and objects. FOAF ontology is a popular ontology for the representation of personal information and social relationships among users.

The location dimension is easy to capture. To be conform with other dimension instantiated, we exploit the existing locations in used collections (for other dimensions). For example, the property *foaf:based_near* in FOAF; the field *location* in BookCrossing. Physical and network contexts are easy to obtain.

Resource. We need a diverse set of resources: containing images, text and video with associated contexts. In the best of our knowledge, there is no collection

[1] `http://online-presence.net/opo/spec/`
[2] `http://www.w3.org/TR/owl-time/`
[3] `http://xmlns.com/foaf/0.1/`

of this type. For this, we define a process to construct a contextual resource collection. This process is based upon two steps detailed in the following:

1. **Step 1: Collecting resources** The purpose of this step is to create a resources collection containing documents of different types. For this, we combine multiple collections of different types. As images and videos collection, we used *FreindFeed*. It is a network that brings together on one page different sources: blog, images, social networking, news, videos, etc. As a text collection, we used *BookCrossing*. BookCrossing is defined as the practice of leaving a book in a public place to be picked up and read by others.
2. **Step 2: Resources contextualization** After collecting necessary resources, we will proceed to a step of *resources contextualization*. Each document would be marked up with a context in a way that can be easily matched against the fields of the user's context. First, we specify the manufacturing context of documents. It specifies the circumstances related to the creation of the document: document author, creation date, number of pages (book resource case), dimensions (image and book resources case), the duration (video resource case), etc. Second, we indicate the editorial context. It is the publication home in the case of a book and articles source in the case of documents from FriendFeed. Third, for each resource, you create a domain (*i.e*, thematic) using *ODP* and *Dbpedia* ontologies.

3.2 Evaluation

GCAPM is a generic model that can be useful to perform any task in P2PIR (recommendation, aggregation, filtering, expansion, etc). In this paper, we present its usefulness in context-aware query expansion task.

Environment. To evaluate the approach proposed in this paper, we built a contextual collection as follows:

- Resources collection contains diversified resources (text, image, video) with their corresponding contexts as described in the previous section.
- Queries collection: we built a queries collection from AOL[4] query log. AOL query log includes more than 30 million web queries collected from more than 650000 users [4]. AOL gives accessed to anonymous users search history specifying the query sent, the sending time and the document clicked by the user with his rank.
- Users collection: We built users collection as described in the previous section using FOAF, BookCrossing and FreindFeed. After the construction of interest centers, a step of queries assignment to users is necessary. To do this, we calculate the similarity between the centers of interest and AOL queries responses. If the similarity is greater than a threshold, we assign this query to the user.

[4] `http://www.gregsadetsky.com/aol-data/`

Results and Discussion. Different test scenarios can be achieved to detect the effectiveness of query expansion that integrates different context dimensions. In this paper, we consider the following scenario: We consider in this case the effectiveness evaluation of context-aware query expansion that integrates the user interests centers. To evaluate the impact of query expansion based on interest centers, we use recall and precision as evaluation metrics. To do so, we computed the average recall per interval of 200 queries. Figure 2 shows that recall increases after query expansion based on interest centers for all queries intervals.

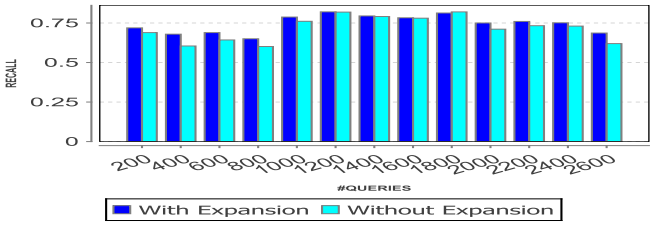
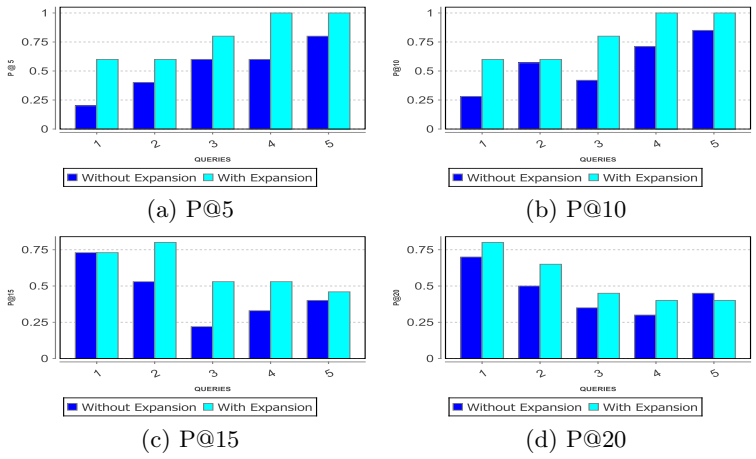


Fig. 2. Variation of Recall



(a) P@5 (b) P@10

(c) P@15 (d) P@20

Fig. 3. Variation of Precision @ k

We observe that for some queries, recall values, in two cases, are close. We decided to observe these queries. Indeed, this is justified by the number of query terms. If the number of query terms is very small, then there is an observable recall improvement and thus expansion is recommended. However, when the number of terms is enough high, the expansion does not have a great effect. Therefore, to better observe the improvement in precision, we chose some short queries (5 queries) and we calculate precision @ k ($k = 5, 10, 15, 20$). As shown in the figure 3, the precision is significantly improved with queries expansion.

4 Conclusion and Future Work

This paper describes our approach for establishing GCAPM. The goal of this proposal is to study how to consider some dimensions in order to provide a generic user model. The main contribution is the definition of a user model that can be used as to model users in P2P systems. We illustrate the use of GCAPM model in the context of query expansion.

In our future work, we are planning to evaluate results agreggation using GCAPM, to exploit the model in existing recommender systems and evaluate the impact in recommendations and the final user satisfaction.

References

1. Bouidghaghen, O., Tamine, L., Boughanem, M.: Context-aware user's interests for personalizing mobile search. In: 12th International Conference on Mobile Data Management, USA, pp. 129–134 (June 2011)
2. Lu, J., Callan, J.: User modeling for full-text federated search in peer-to-peer networks. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, USA, pp. 332–339. ACM (August 2006)
3. Mobasher, B., Dai, H., Luo, T., Sun, Y.: Integrating web usage and content mining for more effective personalization. In: Proceedings of the 1st International Conference on Electronic Commerce and Web Technologies, London, pp. 165–176 (2000)
4. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: Proceedings of the 1st International Conference on Scalable Information Systems, New York (2006)
5. Vassileva, J.: Supporting peer-to-peer user communities. In: Meersman, R., Tari, Z. (eds.) CoopIS/DOA/ODBASE 2002. LNCS, vol. 2519, pp. 230–247. Springer, Heidelberg (2002)
6. Viviani, M., Bennani, N., Egyed-Zsigmond, E.: A survey on user modeling in multi-application environments. In: The Third International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, pp. 111–116 (August 2010)
7. Ye, J., Li, J., Zhu, Z., Gu, X., Shi, H.: Pcsm: A context sharing model in peer-to-peer ubiquitous computing environment. In: Proceedings of the 2007 International Conference on Convergence Information Technology, USA, pp. 1868–1873. IEEE Computer Society (November 2007)
8. Yeferny, T., Bouzeghoub, A., Arour, K.: Context-aware routing method for p2p file sharing systems over manet. In: IMMoA, pp. 21–25 (2012)
9. Yeung, K.F., Yang, Y., Ndzi, D.: A proactive personalised mobile recommendation system using analytic hierarchy process and bayesian network. J. Internet Services and Applications, 195–214 (2012)

Metadata Anchoring for Source Code: Robust Location Descriptor Definition, Building and Interpreting

Karol Rástočný and Mária Bieliková

Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova, Bratislava, Slovakia
{name.surname}@stuba.sk

Abstract. Metadata of dynamic data such as source code have to be maintained after modifications. The first step of metadata maintenance is a repair of metadata anchoring after a modification of the content. Many anchoring approaches exist for regular texts or web pages, but they are not directly applicable for source code. We propose metadata anchoring approach for source code. Metadata represented as information tags are connected to particular line of source code or word in program method or source code file. Our proposal contains a definition of the robust location descriptor and the algorithm for building and interpreting the descriptor. We evaluate our approach on the dataset of change sets from commercial projects with more than sixty thousand C# files.

Keywords: Information tag, anchoring, location descriptor, source code.

1 Introduction

Metadata make human-readable content of files accessible for machines. In case of webpages, metadata are written directly in webpages' content, obviously as XHTML attributes that mark XHTML elements (e.g., RDFa). This way they also interconnect webpages with the Linked Data and enrich webpages with semantics [1]. But solutions that store metadata directly in files' content are not applicable for each file types. Therefore metadata are obviously assigned to whole files and not directly to fragments of their content. We proposed information tags [2], metadata with semantic relation to tagged content, that can be useful not only in web-based documents. We utilize information tags mainly in domain of software development for describing software artifacts by content based features, collaboration metadata and manually assigned features (e.g., a source code ranking assigned by programmers, see Fig. 1).

Information tags are stored separately from target files and they reference tagged fragments by anchors. Concept of information tags has been inspired by human annotations, for that many anchoring approaches has been proposed already. But these anchoring approaches are specialized to specific type of content, often general texts or webpages and they are not properly useable for source code. In this paper we analyze existing anchoring approaches, discuss differences between general texts and source code and propose and evaluate our anchoring approach for source code.

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 372–379, 2013.

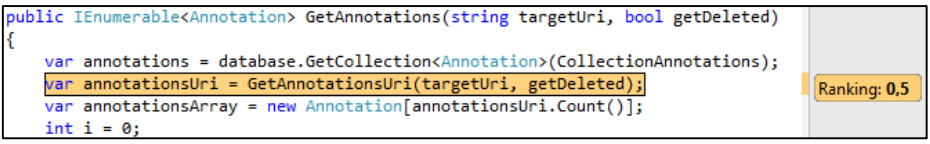


Fig. 1. Visualization of ranking information tag anchored to a source code, provided by extension for Microsoft Visual Studio 2010, developed as part of the research project PerConIK (Personalized Conveying of Information and Knowledge: http://perconik.fiit.stuba.sk/) [3]

2 Textual Anchoring Approaches

The problem of anchoring is the easiest to solve in static texts. Anchoring approaches for texts need not deal with any changes in target content so simple index-based location descriptors are often enough. These descriptors have small memory complexity and their interpretation is fast and simple. A crucial problem arises when modifications of a textual content are uninformed. In these cases, anchoring approaches should use robust locations descriptors and algorithms that can build and interpret such location descriptors. These location descriptors are mainly based on texts, those positions they describe. Main differences between respective robust anchoring approaches are in sizes of location descriptors (amount of information stored in location descriptors), complexity (time and memory complexity of algorithms for building and interpreting location descriptors) and fuzziness (tolerance to size of modifications in documents).

Annotator™ uses as location descriptor a hash value of the smallest unique substring of an annotated text [4]. This approach uses small descriptors that allow fast creation and interpretation, but positions of anchors have to be founded exactly and even a small change in the target text cause, that anchor could not be founded.

A problem of small changes in target texts is partially solved by architecture ComMentor [5] which stores keywords of annotated texts in location descriptors. This has negative influence to sizes of location descriptors and time complexity of algorithms. But small changes in target texts have no influence to validity of anchors.

Both approaches that are used by Annotator™ and ComMentor have another negative aspect – if a target document contains several phrases that are same as a target phrase, there is no mechanism for making a decision on which one phrase is the target phrase. For this reason robust anchoring approaches use location descriptors enriched by contextual information. As one of the simplest contextual information is used an occurrence of a target phrase in a document (realized in the first version of collaborative educational system Alef [6]). This small contextual information is enough when no phrase same as a target phrase is added or modified before the target phrase.

To solve a problem of location descriptors' robustness, Phelps and Wilensky analyzed documents' modifications and defined criteria that have to be fulfilled by a robust descriptor [7]. They also proved an existence of such location descriptor by proposition of a location descriptor, which combines information about an identifier of a target element from SGDOM (simple, general document object model), a path in a SGDOM tree to a target element and a context of an anchor (a text surrounding the target text) [7]. Ideas of the location descriptor were partially adopted by many ap-

proaches, e.g. by Brush et al. [8], who used surrounded texts as contexts and Annotea [9], which utilizes webpages' DOM (document object model). Brush's approach also allows fuzzy interpretation of location descriptors, when unmatched texts from location descriptors are searched by gradual cutting words off front and back of texts.

3 Anchoring Approach for Source Code

Source code of software artifact as one type of texts has several specific properties that should be utilized to propose an anchoring approach which will be more tolerant to modifications in targets with smaller memory and time complexity. To identify these characteristics we have analyzed dataset of 62,115 C# source code files with their revisions from real commercial projects. We have calculated statistics about lines and words and about size of changes over the source code files and methods of classes defined in the files (Table 1).

We decided to calculate statistics for whole files and also for methods of classes defined in the files because we have possibility to anchor information tags to whole files but also to a specific element of AST (abstract syntax tree) of source code. Advantage of anchoring to the whole file is in simple identification of new location of a target text in case when the target text has been moved to another method within the file. It should be also advantageous to pre-process whole file and calculate with pre-processed data over the whole file. On the other side in case of anchoring within a method we have to deal with markedly smaller size of data (see Table 1) and there is smaller chance of conflicts with other positive matches of anchors.

Table 1. Statistics (average values) calculated over 62,115 source code files

Statistics	Per file	Per method
Lines	151.88	15.02
Words	825.64	99.69
Unique lines	80.98	11.91
Unique words	133.18	32.56
Line's length	47.21	40.47
Word's length	7.88	6.44
Lines added	2.37	–
Lines moved	1.29	–
Lines modified	1.26	–
Lines deleted	0.84	–

A source code can be from a general text easily recognized by its structure. While the source code is structured with white characters at beginning of lines and empty lines between logical parts, the general text (e.g., a newspaper article) is mainly comprehensive text, which uses minimal number of white characters to separate words and paragraphs. Upon this observation we can markedly decrease a number of processed characters by source code pre-processing, which replaces sequences of white characters with a single white character.

The next observation is about structural elements of general texts. Lines and words (statements) are smallest elements that give meaning to source code and because their small lengths and relatively stable positions in texts, both of them can be simply used as anchors. Lines of general texts are not important and they have minimal influence to meaning of texts. They also could not be used as anchors because a number of lines and their positions and content are dependent on viewers and their settings.

The last difference which we utilize in our anchoring approach is variety of words in a source code. Due to exactness and formality of a source code, programmers use quite small number of words. In our dataset programmers used in average 826 words per one file, from that only 133 were unique with average length 8 characters (see Table 1). The top story of BBC (http://www.bbc.co.uk/news/business-21933473) with similar length (817 words) has almost three times more unique words (373). If we use words as building element of location descriptor we have to make nine times less word to word comparisons and we can effectively cache results of comparisons.

3.1 Location Descriptor

To deal with problem of anchoring in a source code we proposed an approach, which works with source code files like with sequences of textual elements (lines or words). We propose location descriptor within a target (source file or an AST element of a source file), which allows direct comparisons of location descriptors, identification of tagged texts without target source code file and maintainability of location descriptors without necessity of source code files' change sets, while the first two requirements have to be fulfilled to make information tags suitable for reasoning.

The proposed location descriptor consists of two partial descriptors with different scope of use cases. The first is *index-based location descriptor* which contains indexes of the first and the last letter from a tagged text in a target (see Table 2). This simple descriptor is used in cases when source code file has not been changed or the anchor has exact match at location specified by indexes. The second part is *context-based location descriptor* which is used as the robust location descriptor and it is utilized for determining new location of information tags after modification of source code files. It is also used in cases when data tasks do not need to know exact locations of information tags and they only need to know tagged text or contexts of tagged texts. A context-based location descriptor contains information about *tagged text* as a sequence of textual elements, which has been tagged in a target and about *context before/after tagged text* as minimal unique sequence of at least three textual elements that are directly before/after tagged text (see Table 2).

Table 2. An example of location descriptor for the ranking information tag from the Fig. 1 with words as building elements that are separated by a single white space

Indexes	[168; 230]
Context before	CollectionAnnotations) ;
Tagged text	var annotationsUri = GetAnnotationsUri (targetUri , getDeleted)
Context after	var annotationsArray =

3.2 Interpreting Context-Based Location Descriptors

We interpret context-based location descriptors as sequences of textual elements. The approach gives us opportunity to break up problem of complexity of approximate string matching [10] to two smaller parts – comparing textual elements and local sequence alignment. So we radically decrease size of data that have to be processed in one step. Whole algorithm of interpreting context-based location descriptors calculate index-based location descriptors of matched locations with their confidences:

1. *Pre-process a source code* – the source code is spitted to textual elements;
2. *Compare textual elements* – unique textual elements of the source code are compared to unique textual elements of a context-based location descriptor by a string similarity algorithm;
3. *Local sequence alignment* – locations of a tagged text are searched as subsequences of the source code via the Smith-Waterman algorithm (optimal local sequence alignment algorithm [11]). In this step we calculate scores for matched location via (1), where i (an index of the last matched element in source code) and j (an index of the last matched element of the tagged text) are indexes in scoring matrix H, for that a match has been found, $Sc(i,j)$ is a score for the match, $H_{i,j}$ is value in scoring matrix for the match and $seq_{i,j}$ is aligned sequence for the match.

$$Sc(i, j) = \frac{H_{i, j}}{\left| seq_{i, j} \right|} \tag{1}$$

4. *Calculate scores of matches of contexts* – scores of contexts before and after tagged text are calculated for each possible alignment of tagged text. The confidences are calculated via the Smith-Waterman algorithm whose scoring function (1) uses as aligned sequences, sequences connected to alignment of tagged texts, e.g. if context before tagged text is "B C", tagged text is "E F" and source code is "A B C D E F", the scoring function uses as aligned sequence "B C D".
5. *Calculate confidences of matched locations* – confidences of matched locations are calculated as a linear combination of scores for each possible alignment of tagged text, where scores of context before and a tagged text have weight 0.2 and a score of the tagged text has weight 0.6. Weights of scores are designed to reach at least 0.5 confidence after shifting a tagged text to another part of a source code (scores of contexts are 0) with a small modification in the tagged text.

4 Evaluation

Main goals of evaluation of the proposed method are acknowledgement of usability for real-time processing and for accurate precision in source code and measuring precision and time and memory complexity for different parameters. Usability condition of real-time processing is based on necessity of maintenance after each modification in information tags' targets. For this reason we can set same goal for both cases – 50

milliseconds for one interpreting of location descriptor on a single processor's 2.8GHz core (we expect load of maintaining 20 information tags per one second based on size of modifications in a source code file – see Table 1).

Our interpreting algorithm calculates confidences of determined locations in range <0; 1>, while we used threshold 0.5 to determine whether the algorithm matched location or does not. So in ideal case we expect that the algorithm matches right location with confidence 1, while greater confidence is better. In case, when the algorithm does not match right location we prefer matches that overlap right locations, than we prefer cases with no match and the worst cases for us are mismatched locations.

4.1 Dataset and Evaluation Plan

We have built a dataset of modifications in a source code for evaluation of proposed approach (http://perconik.fiit.stuba.sk/public/SourceCodeAnchoringEval.zip). We selected one statistically average source code file and we made multiple modifications in the file, that affect contexts (TC I: Add line before tagged position; TC II: Add line before and after tagged position; TC III: Add three lines before tagged position; TC IV: Add three lines before and after tagged position; TC V: Delete line before tagged position; TC VI: Delete line before and after tagged position; TC VII: Delete three lines before tagged position; TC VIII: Delete three lines before and after tagged position) and a tagged text (TC IX: Misspell correction; TC X: Rename an attribute/method; TC XI: Copy the tagged text four lines before the original position; TC XII: Modify a half of the tagged text).

We work with one and three lines, because in the average programmers add, delete, modify and move from one to three lines in a file (see Table 1) and also because our algorithm uses at least three textual elements as contexts. During the evaluation we measure size of context-based location descriptors, time necessary for building and interpreting location descriptors and confidences of interpreted location descriptors.

We evaluate each test case with different set-ups of configurable parameters – *size of target text* (methods vs. whole source code file), *size of textual element* (words vs. lines) and *string similarity algorithm* used for comparison of textual elements. We use Block distance, Chapman length deviation, Cosine similarity, Dice similarity, Euclidean distance, Jaccard similarity, Jaro, Jaro-Winkler, Levenstein, Matching coefficient, Monge-Elkan, Needleman-Wunch, Overlap coefficient, Smith-Waterman and Smith-Waterman-Gotoh string similarity algorithms. To avoid implementation differences and make the evaluation more rigorous we made decision to not implement these algorithms but to use existing library *SimMetrics* [12], which is used in multiple research projects for its quality (e.g., [13]).

4.2 Results and Discussion

From measured sizes of location descriptors, we can conclude that the size of context-base location descriptors for words as textual elements is more than two times smaller than location descriptors for lines when only one line is tagged. In case of three tagged lines, both words and lines as textual elements reached almost same values. On

the other side, time necessary for building location descriptors is more than three times longer if we use words as textual elements towards set-ups with lines.

All reached sizes of location descriptors and also built times fulfilled our requirements. The other problem is usability of these descriptors after modifications in source code files. All set-ups of the location descriptors' interpret function reached satisfactory results for almost all test cases. The exception is the most complex test case XII, where almost a half of the tagged text was modified. For this test case, best results were reached by Smith-Waterman-Gotoh and Monge-Elkan string similarity algorithms, but Smith-Waterman-Gotoh algorithm had several mismatched locations in test cases IV, VIII and X. Satisfactory results were reached also by Levenstein, Jaro and Jaro-Winkler string similarity algorithms. These algorithms did not match all tagged texts in test case XII, but they had one unmatched location and they had only one mismatched location at a set-up, in which almost all string similarity algorithms had mismatches, too.

The last indicator of a usability of the proposed approach is a time complexity of the interpret function. To evaluate this metric, we compare times consumed by all set-ups with four abovementioned string similarity algorithms. For all test cases, set-ups with Monge-Elkan algorithm needed longest times to interpret location descriptors. This string similarity reached best confidences but times highly exceeded our requirements (50 ms) for longer target texts, what made this algorithm unusable.

Remaining three string similarity algorithms needed almost same times to interpret location descriptors in set-ups where words were used as textual elements. Remarkable differences were measured in set-ups with lines as textual elements. In these cases, Levenstein algorithm needed two or three times longer times. Jaro and Jaro-Winkler algorithms still reached almost same times. But almost in all cases set-ups with Jaro, Jaro-Winkler and Levenstein algorithms consumed less than fifty milliseconds what makes all of these set-ups useable for real-time processing. The only exceptions were cases, in that we searched three tagged lines over whole file with words as textual elements. In these cases all three string similarity algorithms needed around one time longer times than allowed fifty milliseconds.

5 Conclusions

We propose anchoring approach applicable for maintenance metadata of changing source code. As metadata we consider both machine generated information tags (e.g. similarity with code smells) and user generated information tags (e.g. keywords describing code snippets). Information tags provide source for useful information such as identification of bad practices, evaluation of source code quality based on an estimation of the programmer state (e.g., tired, stressed), recommendation of good practices and tricks/snippets used by peers. They also serve as an input for reasoning on properties of software artifacts such as similarity with code smells, estimation of developer skill and proficiency.

The evaluation approved, that the proposed anchoring approach is sufficiently accurate and its time complexity allows its usability for real-time processing. The op-

timal configuration is usage of the Jaro-Winkler string similarity algorithm, while if our anchoring approach has to be used with a longer source code, usage of lines as textual elements of sequences ensures usability for real-time processing.

Acknowledgements. This work was partially supported by the grants VG1/0675/11, APVV-0233-10 and it is the partial result of the Research & Development Operational Program for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

1. Araujo, S., Houben, G.-J., Schwabe, D.: Linkator: Enriching web pages by automatically adding dereferenceable semantic annotations. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G., et al. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 355–369. Springer, Heidelberg (2010)
2. Rástočný, K., Bieliková, M.: Maintenance of Human and Machine Metadata over the Web Content. In: Grossniklaus, M., Wimmer, M. (eds.) ICWE 2012 Workshops. LNCS, vol. 7703, pp. 216–220. Springer, Heidelberg (2012)
3. Bieliková, M., et al.: Webification of Software Development: General Outline and the Case of Enterprise Application Development. In: Procedia Technology, 3rd World Conf.on Information Technology (to appear)
4. Ovsiannikov, I.A., Arbib, M.A., Mcneill, T.H.: Annotation Technology. Int. Journal of Human-Computer Studies 50, 329–362 (1999)
5. Röscheisen, M., Winograd, T., Paepcke, A.: Content Ratings, and Other Third-Party Value-Added Information: Defining an Enabling Platform. Stanford (1997)
6. Šimko, M., Barla, M., Bieliková, M.: ALEF: A Framework for Adaptive Web-Based Learning 2.0. In: Reynolds, N., Turcsányi-Szabó, M. (eds.) KCKS 2010. IFIP AICT, vol. 324, pp. 367–378. Springer, Heidelberg (2010)
7. Phelps, T.A., Wilensky, R.: Robust intra-document locations. Computer Networks 33, 105–118 (2000)
8. Brush, A.J.B., Bargeron, D., Gupta, A., Cadiz, J.: Robust annotation positioning in digital documents. In: Proc.of the SIGCHI Conf.on Human Factors in Comp.Systems, CHI 2001, pp. 285–292 (2001)
9. Kahan, J., Koivunen, M.-R.: Annotea: An Open RDF Infrastructure for Shared Web Annotations. In: Proc. of the 10th Int. Conf.on World Wide Web, WWW 2001, pp. 623–632. ACM Press, New York (2001)
10. Navarro, G.: A guided tour to approximate string matching. ACM Comp. Surveys 33, 31–88 (2001)
11. Akoglu, A., Striemer, G.M.: Scalable and Highly parallel Implementation of Smith-Waterman on Graphics Processing Unit Using CUDA. Cluster Comp. 12, 341–352 (2009)
12. Norton, B., Chapman, S., Ciravegna, F.: Orchestration of Semantic Web Services for Large-Scale Document Annotation. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 649–663. Springer, Heidelberg (2005)
13. Heuser, C.A., Krieser, F.N.A., Orengo, V.M.: SimEval - A Tool for Evaluating the Quality of Similarity Functions. In: Grundy, et al. (eds.) ER 2007 Tutorials, Posters, Panels and Industrial Contributions at the 26th Int. Conf.on Conceptual Modeling, vol. 83, pp. 71–76. Australian Computer Society, Inc., Darlinghurst (2007)

The Price Is Right
Models and Algorithms for Pricing Data

Ruiming Tang[1], Huayu Wu[2], Zhifeng Bao[3], Stéphane Bressan[1], and Patrick Valduriez[4]

[1] School of Computing, [3]IDMI, National University of Singapore
{tangruiming,dcsbz,steph}@nus.edu.sg
[2] Institute for Infocomm Research, Singapore
huwu@i2r.a-star.edu.sg
[3] INRIA & LIRMM, Montpellier, France
Patrick.Valduriez@inria.fr

Abstract. Data is a modern commodity. Yet the pricing models in use on electronic data markets either focus on the usage of computing resources, or are proprietary, opaque, most likely ad hoc, and not conducive of a healthy commodity market dynamics. In this paper we propose a generic data pricing model that is based on minimal provenance, i.e. minimal sets of tuples contributing to the result of a query. We show that the proposed model fulfills desirable properties such as contribution monotonicity, bounded-price and contribution arbitrage-freedom. We present a baseline algorithm to compute the exact price of a query based on our pricing model. We show that the problem is NP-hard. We therefore devise, present and compare several heuristics. We conduct a comprehensive experimental study to show their effectiveness and efficiency.

1 Introduction

Data has value. Brynjolfsson et al. remark that "Organizational judgement is currently undergoing a fundamental change, from a reliance of a leader's human intuition to a data based analysis" ([3]). They conduct a study of 179 large publicly traded firms showing that those adopting data-driven decision-making increase output and productivity beyond what can be explained by traditional factors and by IT usage alone. In the science community, data is taking such a prominent place that Jim Gray argues that "The techniques and technologies for such data-intensive science are so different that it is worth distinguishing data-intensive science from computational science as a new, *fourth paradigm* for scientific exploration." [8]

As argued by the authors in [2,11], data is being bought and sold. Electronic data market places are being designed and deployed. Independent vendors, such as Aggdata [13] and Microsoft's Azure MarketPlace [14], aggregate data and organize their online distribution. A lot of effort has been made to define computing and resource based pricing models ([7], for instance). Yet, only recently have authors looked at pricing data independently from computation beyond

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 380–394, 2013.

the basic pricing models found in current commercial data services where buyers are restricted to buy data by the volume or through pre-defined views. Such pricing models are simplistic, inflexible and can create undesirable arbitrage situations (i.e. when a sought result can be obtained at a cheaper price by issuing and combining the results of several queries that by issuing a single and direct query) [11].

In response to this observation, the authors of [11] propose a pricing model that defines the price of an arbitrary query as the minimum sum of the prices of views that can determine the query. The model is flexible since it explicitly allows the combination of views while preventing arbitrage. Unfortunately, the authors do not propose generic algorithms for the computation of prices in their model. In addition, the authors assign prices to individual pre-defined views, however, the view granularity might be too coarse for many applications. Even though, in principle, the view model can degenerate to a tuple model in which each tuple is a view, such an approach raises serious scalability issues. Hence we adapt this pricing strategy for a tuple granularity, i.e. we charge for a query based on the source tuples used to answer this query. Therefore we need to track the source tuples which are used to produce query result, and data provenance is well known for this purpose. In this paper, we devise a pricing model with a tuple granularity by leveraging and extending the notion of data provenance, i.e. set of tuples contributing to the result of a query.

Our model assigns a price to each source tuple in the database. We make sure that each tuple is charged for only once even if it contributes in the query result multiple times, due to the nature of information products, i.e. when an information product is used to produce an output, it is not consumed, instead, it still can be used to produce other outputs. For this reason, we need to devise a more rigorous provenance model than the existing ones (e.g. [6,4,9]). Moreover, we propose a range of price aggregation methods, using p-norm [16], that give the service providers the possibility to tune and adapt their pricing strategies. Our pricing model fulfils the following desirable properties:

Contribution monotonicity: the more tuples one query uses, the higher its price is.

Bounded-price: the price of a given query is not higher than the price of all the tuples in the relations involved in the query.

Contribution arbitrage-freedom: the price of a query Q cannot be higher than the sum of the prices of queries, the union of whose contributing tuples is a superset of the contributing tuples of Q.

Finally, we devise algorithms for the computation of the price of data in this model. As we will show, in general, computing the exact prices for queries is intractable, but we devise several generic algorithms using different heuristics. We evaluate their performance and scalability and compare them to those of a baseline algorithm.

2 Background and Related Work

2.1 Relational Data Provenance Semantics

The authors of [5] state that there are three common forms of "provenance" in relational databases. The first form is "where-provenance" [4], targeting at the attribute granularity and showing where the values of the attributes of a result tuple come from. The second form is "why-provenance", targeting at the tuple granularity and returning the source tuples that explain why a result tuple is in the result. [6] and [4] introduce two kinds of "why-provenance". The other form is "how-provenance" [9], targeting at tuple granularity and showing that how a result tuple is produced by source tuples.

The most related provenance to our model is the one introduced by [4]. The authors of [4] define the notion of why-provenance (named witness basis in [4]) in terms of a deterministic semistructured data model and query language. The authors of [5] adapt the definitions in [4] to the relational model and relational algebra. Among the set of why-provenances, they define "minimal" ones. **A why-provenance is minimal (called minimal why-provenance) if and only if none of its proper subsets can produce the result tuple**. There may exist several minimal why-provenances for a result tuple.

The existing provenances work for individual result tuples. If we use them directly, we have to price a query by computing the price of each result tuple separately based on its provenance, and summing them up. Its shortcoming is that a source tuple may be charged more than once, which violates the nature of information products, i.e. when an information product is used to produce an output, it is not consumed, instead, it still can be used to produce other outputs. Therefore we have to make sure that each tuple is charged for only once even if it contributes in the query result multiple times. For this reason, we need to devise a more rigorous provenance model which works for a set of tuples.

2.2 Data Pricing

The authors of [15] survey existing pricing strategies such as free, usage based prices, package pricing, flat fee tariff, two-part tariff and freemium. However, the authors of [2], argue that these pricing models suffer from several weaknesses, such as assuming all tuples are of equal value, leading to arbitrage situations. There exist some works focusing on setting prices to information goods to get maximum profit, such as [18]. Some works (e.g. [7]) aim to charge for services and resources in the context of cloud computing.

Different from other works, the authors of [11] propose a pricing model charging for individual queries. Their query-based pricing model allows the seller to set explicit prices for only a few views; the price of a query is the price of the cheapest set of views which can determine the query.

In this paper, our framework also prices queries but differentiates itself from the framework of [11] in the following aspects. **Firstly**, our pricing model charges for a query based on the source tuples needed to answer this query, which means

that we target at the tuple granularity, while the model in [11] targets at the view granularity. The view granularity might be too coarse for many applications. Even though, in principle, the view model can degenerate to a tuple model in which each tuple is a view, such an approach raises serious scalability issues. **Secondly**, We propose an exact algorithm and several approximation algorithms to compute prices of queries using our pricing function while [11] does not.

3 Pricing Relational Data

In this section, we propose a generic pricing model which consists of a price setting function and a pricing function in Section 3.1. Later in Section 3.2, we define the set of minimal provenance(s) of the result of a query, which is invariant under query rewriting. We use p-norm to define the price of a set of tuples in Section 3.3. In Section 3.4, we propose a pricing function, that defines the price of a query solely based on its *contributing tuples*.

3.1 Pricing Model

Our pricing model consists of two functions. One of them is used by the data seller to assign a price to each source tuple, and the other is used by the seller to define the prices of input queries.

Definition 1. *(Price setting function and pricing function). Let $\mathcal{D}$ be a database. A* price setting function *is a function $s_{\mathcal{D}} : \mathbb{T} \to \mathbb{R}^+$, where $\mathbb{T}$ is the set of source tuples in $\mathcal{D}$, and $\mathbb{R}^+$ is the set of non-negative real numbers. A* pricing function *is a function $pr_{\mathcal{D}} : \mathbb{Q} \to \mathbb{R}^+$, where $\mathbb{Q}$ is the set of queries, and $\mathbb{R}^+$ is the set of non-negative real numbers.*

3.2 Minimal Provenance of a Set of Tuples

We define a provenance of the query result as a whole, instead of using the provenance of individual resulting tuples (as mentioned in Section 2.1). We consider necessary and sufficient tuples to produce the result of a given query to be a minimal provenance of the query result.

Definition 2. *(Provenance of a set of tuples). Let Q be a query. Let $\mathcal{D}$ be a database. Let $Q(\mathcal{D})$ be the result of the query Q on the database $\mathcal{D}$. A* provenance *of $Q(\mathcal{D})$ is a set of tuples L ($L \subseteq \mathcal{D}$) such that $Q(\mathcal{D}) \subseteq Q(L)$*[1]*.*

Definition 3. *(Minimal provenance of a set of tuples). A* minimal provenance *of $Q(\mathcal{D})$ is a provenance L of $Q(\mathcal{D})$ such that $\forall L', L' \subseteq L \Rightarrow L' = L$ where L' is a provenance of $Q(\mathcal{D})$.*

[1] If Q is a monotonic query (which contains only selection, projection, join and union) it is only possible that $Q(\mathcal{D}) = Q(L)$. For a non-monotonic query Q, it is possible that $Q(\mathcal{D}) \subseteq Q(L)$.

Example 1. $Q(\mathcal{D}) = \{t_1, t_2\}$. The sets of minimal why-provenance of t_1 and t_2 are $\{\{x_1, x_5\}, \{x_1, x_6\}\}$ and $\{\{x_2, x_7\}, \{x_4, x_8\}\}$, respectively. There are four minimal provenances of the query result $Q(\mathcal{D})$: $L_1 = \{x_1, x_2, x_5, x_7\}$, $L_2 = \{x_1, x_4, x_5, x_8\}$, $L_3 = \{x_1, x_2, x_6, x_7\}$, $L_4 = \{x_1, x_4, x_6, x_8\}$. The semantics of L_1 is that the set of tuples $\{x_1, x_2, x_5, x_7\}$ produces the query result, but none of any subsets of L_1 is able to produce the query result. □

From Example 1, we can see that the query result $Q(\mathcal{D})$ may have several minimal provenances. We use $M(Q, \mathcal{D})$ to represent the set of minimal provenances of $Q(\mathcal{D})$.

Theorem 1. *Two equivalent queries have the same set of minimal provenances for any database.*

Theorem 1 states that the set of minimal provenances remains invariant under query rewriting. The proof of Theorem 1 is in our technical report [17].

3.3 *p*-Norm

In this section, we use p-norm to define the price of a set of tuples. The p-norm [16] is a function defining a norm of a vector.

Definition 4. *(p-norm of a vector). Let X be a vector $(x_1, x_2, ..., x_n)$, for a real number $p \geq 1$, the p-norm of X is defined by $||X||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$.*

Although Definition 4 defines the p-norm of a vector, this definition can be easily adapted to be the p-norm of a set.

Proposition 1. *If $p \geq 1$ and $a \geq 0$, $||X||_{p+a} \leq ||X||_p$.*

Proposition 1 is proved in [16]. It states the fact that the p-norm value decreases as p ($p \geq 1$) increases.

Definition 5. *(The price of a set of tuples). Let $\mathcal{D}$ be a database. Let X be a set of tuples, and $X \subseteq \mathcal{D}$. Let $s_{\mathcal{D}}$ be a price setting function. The price of the set of tuples X is defined by $||X||_p = (\sum_{i=1}^{n} s_{\mathcal{D}}(t_i)^p)^{\frac{1}{p}}$, where $t_i \in X$.*

Using p-norm gives the data seller the possibility to tune and adapt their pricing strategies. Due to Proposition 1, Definition 5 allows the seller to define the price of a set of tuples as a function that ranges from a sum of the prices of the individual tuples when p is equal to 1, to a maximum price of the individual tuples when p tends to infinity. In between, the value of the price decreases as p increases. p-norm would be useful and convenient when the data seller wants to give different discounts to different categories of consumers.

3.4 The Pricing Function

Our pricing function defines the price of a query based on the necessary and sufficient tuples to produce the result. We start with defining such tuples. After that, we define several required properties that a pricing function has to satisfy. At last, we propose our pricing function which satisfies all the properties.

Definition 6. *(Contributing tuples). Let Q be a query. Let $\mathcal{D}$ be a database. Let $Q(\mathcal{D})$ be the result of the query Q on the database $\mathcal{D}$. The set of* contributing tuples *of $Q(\mathcal{D})$ is the set of minimal provenances of $Q(\mathcal{D})$, i.e. $M(Q, \mathcal{D})$.*

Property 1: Contribution monotonicity. Intuitively, a query using more tuples should have a higher price than another query using less tuples. However, it is not clear what is the meaning of "using more tuples" when there exist several ways to produce the query result. Informally, we say that a query Q_2 uses more tuples than another query Q_1 if for every set of tuples to produce the result of Q_2, it is possible to produce the result of Q_1 only using its subset.

Definition 7. *(Contribution containment). Let $\mathcal{D}$ be a database. Let Q_1 and Q_2 be two queries. $M(Q_1, \mathcal{D})$, $M(Q_2, \mathcal{D})$ are the sets of contributing tuples of $Q_1(\mathcal{D})$, $Q_2(\mathcal{D})$, respectively. Q_1 is* contribution contained *in Q_2 with respect to $\mathcal{D}$, namely $Q_1 \subseteq_{C(\mathcal{D})} Q_2$, if and only if:*

$$\forall L'(L' \in M(Q_2, \mathcal{D}) \Rightarrow \exists L(L \in M(Q_1, \mathcal{D}) \wedge L \subseteq L'))$$

Example 2. Let $\mathcal{D}$ be a database. Let Q_1, Q_2 and Q_3 be three queries. $M(Q_1, \mathcal{D})$, $M(Q_2, \mathcal{D})$ and $M(Q_3, \mathcal{D})$ are the sets of contributing tuples of $Q_1(\mathcal{D})$, $Q_2(\mathcal{D})$ and $Q_3(\mathcal{D})$. Assume $M(Q_1, \mathcal{D}) = \{l_1^1 = \{t_1, t_2\}, l_1^2 = \{t_2, t_3\}\}$, $M(Q_2, \mathcal{D}) = \{l_2^1 = \{t_1, t_2, t_3\}\}$, $M(Q_3, \mathcal{D}) = \{l_3^1 = \{t_1, t_2\}, l_3^2 = \{t_1, t_3\}\}$.

$Q_1 \subseteq_{C(\mathcal{D})} Q_2$ because for l_2^1, there exists l_1^1 such that $l_1^1 \subseteq l_2^1$. However, $Q_1 \nsubseteq_{C(\mathcal{D})} Q_3$ because for l_3^2, $l_1^1 \nsubseteq l_3^2$ and $l_1^2 \nsubseteq l_3^2$. □

The price of Q_1 should be lower than the price of Q_2 if for every way to produce the result of Q_2, we can find a cheaper way to produce the result of Q_1.

Definition 8. *(Contribution monotonicity). A pricing function is said to be* contribution monotonic *if given a database $\mathcal{D}$, whenever two queries Q_1, Q_2 satisfy $Q_1 \subseteq_{C(\mathcal{D})} Q_2$, their prices satisfy $pr_{\mathcal{D}}(Q_1) \leq pr_{\mathcal{D}}(Q_2)$.*

Property 2: Bounded-price. Since we charge for each source tuple in the database at most once, it is reasonable that the upper bound of the price of any query is the price of the entire database, or more precisely, is the price of the source tuples in the relations involved in the query. We call this property *bounded-price*, defined as follows.

Definition 9. *(Bounded-price). A pricing function is said to be of* bounded-price *if for any database $\mathcal{D}$, the price of a query is always not higher than the price of the source tuples in the relations which are involved in the query, i.e. $pr_{\mathcal{D}}(Q) \leq ||S||_p$, where $S = \{t \in \mathcal{R} | \mathcal{R} \propto Q\}$ and $\mathcal{R} \propto Q$ means that the relation $\mathcal{R}$ is involved in the query Q.*

Lemma 1. *If a pricing function is contribution monotonic, it is of bounded-price.*

Lemma 1 shows that bounded-price is implied by contribution monotonicity. The detailed proof is available in our technical report [17].

Property 3: Contribution arbitrage-freedom. In economics, arbitrage is possible when equivalent assets are available for two different prices ([1]). In [11], arbitrage refers to the case that it might turn out to be less expensive to issue several queries and combine their results than issuing a single query.

Our concern is contributing tuples, therefore we have to adapt the concept of arbitrage for contributing tuples. Consider a query Q with the set of contributing tuples $\{\{t_1, t_2, t_3\}\}$ (with the price p), another query Q_1 with the set of contributing tuples $\{\{t_1, t_2\}\}$ (with the price p_1), and a third query Q_2 with the set of contributing tuples $\{\{t_3, t_4\}\}$ (with the price p_2). The data seller would ensure that $p < p_1 + p_2$. Otherwise, it results in the price of $\{t_1, t_2, t_3, t_4\}$ being lower than the price of $\{t_1, t_2, t_3\}$, which is arbitrage and is not reasonable.

Informally, contribution arbitrage-freedom tells that the price of a query Q cannot be higher than the sum of the prices of queries, the union of whose contributing tuples is a superset of the contributing tuples of Q. However, in general, for each query, there may exist more than one way to produce the query result. We require that the price of Q is not higher than the sum of the prices of Q_i if for the union of every combination of minimal provenances of Q_i, there exists a minimal provenance of Q to be a subset of the union set.

Definition 10. *(Contribution arbitrage-freedom). Let $\mathcal{D}$ be a database, Q be a query, and $\{Q_i\}(1 \leq i \leq m)$ be a set of queries. $M(Q, \mathcal{D})$ is the set of contributing tuples of query Q, and $M(Q_i, \mathcal{D})$ is the set of contributing tuples of query Q_i. A pricing function is* contribution arbitrage-free *if*

$$(\forall S(S = \bigcup_{1 \leq i \leq m} L^i_{k_i}(L^i_{k_i} \in M(Q_i, \mathcal{D})) \Rightarrow \exists L(L \in M(Q, \mathcal{D}) \wedge L \subseteq S)))$$

then $\sum_i pr_{\mathcal{D}}(Q_i) \geq pr_{\mathcal{D}}(Q)$ holds.

Example 3. Let $\mathcal{D}$ be a database. Let Q_1, Q_2 and Q be three queries. $M(Q_1, \mathcal{D})$, $M(Q_2, \mathcal{D})$ and $M(Q, \mathcal{D})$ are the sets of contributing tuples of $Q_1(\mathcal{D})$, $Q_2(\mathcal{D})$ and $Q(\mathcal{D})$. Assume $M(Q_1, \mathcal{D}) = \{l^1_1 = \{t_1, t_2\}, l^2_1 = \{t_2, t_3\}\}$, $M(Q_2, \mathcal{D}) = \{l^1_2 = \{t_2, t_4\}, l^2_2 = \{t_1, t_4\}\}$, $M(Q, \mathcal{D}) = \{l^1 = \{t_1, t_2\}, l^2 = \{t_2, t_3\}, l^3 = \{t_1, t_4\}\}$.

For $l^1_1 \cup l^1_2 = \{t_1, t_2, t_4\}$, there exists $l^1 \subseteq l^1_1 \cup l^1_2$. For $l^2_1 \cup l^1_2 = \{t_2, t_3, t_4\}$, there exists $l^2 \subseteq l^2_1 \cup l^1_2$. For $l^1_1 \cup l^2_2 = \{t_1, t_2, t_4\}$, there exists $l^1 \subseteq l^1_1 \cup l^2_2$. For $l^2_1 \cup l^2_2 = \{t_1, t_2, t_3, t_4\}$, there exists $l^1 \subseteq l^2_1 \cup l^2_2$.

If we have $pr_{\mathcal{D}}(Q_1) + pr_{\mathcal{D}}(Q_2) \geq pr_{\mathcal{D}}(Q)$, this pricing function is contribution arbitrage-free. □

Definition 11. *(Price of a query). The price of a query Q in a database $\mathcal{D}$ is defined as the price of the cheapest minimal provenance of $Q(\mathcal{D})$:*

$$pr_{\mathcal{D}}(Q) = min_{L \in M(Q, \mathcal{D})} ||L||_p$$

Theorem 2. *The pricing function in Definition 11 is contribution monotonic, of bounded-price and contribution arbitrage-free. Moreover, The prices are the same among equivalent queries.*

The detailed proof of Theorem 2 is available in our technical report [17].

4 Computing Price

In general cases, computing the exact price of a given query is NP-hard. We first present an algorithm to compute the exact price in Section 4.1. Then we present four approximation algorithms with different heuristics in Section 4.2.

4.1 Exact Algorithm

Theorem 3. *Given the price of each source tuple in database $\mathcal{D}$, the query result $Q(\mathcal{D})$, and the minimal why-provenances of each result tuple $t_i \in Q(\mathcal{D})$, computing the price $pr_{\mathcal{D}}(Q)$ is NP-hard.*

We prove this theorem by reducing this problem to the MinCostSAT problem ([12]), which is NP-hard. Detailed proof can be found in our technical report [17].

In this section, we present a baseline algorithm to compute the exact price of a query using our proposed pricing function, though the complexity of the algorithm is potentially exponential.

The key problem is, given a query Q, and a database $\mathcal{D}$, how to compute the set of minimal provenances $M(Q, \mathcal{D})$. We have stated the difference between our proposed minimal provenance and the minimal why-provenance ([4]) in the related work. Here we study the relationship between them in detail.

We denote the set of minimal provenances of the query result $Q(\mathcal{D})$ by $M(Q, \mathcal{D})$. For a tuple $t_i \in Q(\mathcal{D})$, we denote its set of minimal why-provenances ([4]) by W^i. We denote the set in which each element is the union of a minimal why-provenance of each result tuple by $U(Q, \mathcal{D})$, i.e. $U(Q, \mathcal{D}) = \{\bigcup_i w^i_{k_i} | w^i_{k_i} \in W^i\}$. Using the example below, we verify that $M(Q, \mathcal{D}) \neq U(Q, \mathcal{D})$.

Example 4. Assume $Q(\mathcal{D}) = \{t_1, t_2\}$. The sets of minimal why-provenances of t_1 and t_2 are assumed to be $\{\{a_1, a_2\}, \{a_1, a_4\}\}$ and $\{\{a_3, a_4\}\}$, respectively. $U(Q, \mathcal{D}) = \{\{a_1, a_2, a_3, a_4\}, \{a_1, a_3, a_4\}\}$. It is not the set of minimal provenances. More specifically, $\{a_1, a_2, a_3, a_4\}$ cannot be a minimal provenance, since its proper subset $(\{a_1, a_3, a_4\})$ is a minimal provenance. □

Example 4 indicates that $U(Q, \mathcal{D})$ may include non-minimal provenances which are not included in $M(Q, \mathcal{D})$. We denote the set after filtering all the non-minimal provenances from $U(Q, \mathcal{D})$ by $U'(Q, \mathcal{D})$.

Theorem 4. *Let Q be a query. Let $\mathcal{D}$ be a database. Let $Q(\mathcal{D})$ be the result of the query Q on the database $\mathcal{D}$. It is true that $M(Q, \mathcal{D}) = U'(Q, \mathcal{D})$.*

The proof of Theorem 4 is available in our technical report [17]. According to Theorem 4, in order to get $M(Q, \mathcal{D})$, non-minimal provenances have to be filtered from $U(Q, \mathcal{D})$ (to generate $U'(Q, \mathcal{D})$). However, the "filtering" is very expensive. Assume we have n tuples in the query result $Q(\mathcal{D})$, and each tuple has m minimal why-provenances. In this case, $U(Q, \mathcal{D})$ contains m^n provenances. To check the minimality of a provenance u in $U(Q, \mathcal{D})$, the complexity is $O(m^n)$, since we have to check the inclusion relationship between u and all other $m^n - 1$ provenances.

The minimality of all the provenances in $U(Q, \mathcal{D})$ have to be checked, therefore the complexity is $O(m^{2n})$. However, the expensive "filtering" is not needed. Our aim is not computing $M(Q, \mathcal{D})$, but computing the price of the query, which is the price of the cheapest provenance. If a provenance is not minimal, it cannot be the cheapest provenance, hence it will not affect the price. Therefore it is true that $min_{L \in U(Q,\mathcal{D})}||L||_p = min_{L \in U'(Q,\mathcal{D})}||L||_p$. The detailed reasoning can be found in our technical report [17].

Based on the above observations, we devise a baseline algorithm to compute the exact price of a query. The pseudo code is in our technical report [17]. Its correctness is guaranteed by Theorem 4. Assume we have n result tuples, and each tuple has m minimal why-provenances. Firstly the algorithm computes the set of minimal why-provenances of each tuple, which we assume to get beforehand. Then, it constructs the set of provenances of the query result $U(Q, \mathcal{D})$ with the complexity $O(m^n)$. Lastly, it computes the prices of all the provenances and chooses the lowest one with the complexity $O(m^n)$. In total, the complexity is $O(m^n)$.

4.2 Approximation Algorithms

Since computing the exact price of a query in general cases is intractable, we devise some approximation algorithms to compute approximate price of a given query. Before introducing the heuristics, we define some notations that are used in this section. We denote the number of result tuples by n, the maximum number of minimal why-provenances of result tuples by m and the number of source tuples that may contribute to the query result by b. w_i^{min} represents the minimal why-provenance of t_i that minimizes an objective function (in approximation algorithms). w_i^{min} varies from one heuristic to another.

Heuristics. In this section, we introduce the heuristics and present the approximation algorithms. Before going into details, we first discuss the intuition of the heuristics. Our aim is to get the cheapest provenance of the result of a query, given the set of minimal why-provenances of individual result tuples.

The first heuristic approximates the cheapest provenance of the query result by union of the cheapest minimal why-provenance of each result tuple. This heuristic seeks for individual local optimal values, instead of global optimal value. We refer to this heuristic as Heuristic 1 in Table 1. The second heuristic (Heuristic 2 in Table 1) uses the same intuition as Heuristic 1, but with a different function to choose minimal why-provenances for each result tuple. Instead of choosing the cheapest minimal why-provenance as Heuristic 1, Heuristic 2 chooses the minimal why-provenance with the lowest average price.

Neither of Heuristic 1 and 2 memorizes their previous choices of minimal why-provenances and they choose a minimal why-provenance of each result tuple independently. Based on this observation, we devise another two heuristics (Heuristic 3 and 4 in Table 1). Heuristic 3 and 4 memorize their previous choices of minimal why-provenances when they are choosing a minimal why-provenance

of the next result tuple. The difference between Heuristic 3 and 4 is that the former one chooses the cheapest minimal why-provenance for each result tuple, while the latter one chooses the one with lowest average price.

We summarize the similarities and differences among the four heuristics in Table 1. Due to space limit, we only present the algorithms with Heuristic 1 and 3. The algorithms with Heuristic 2 and 4 are in our technical report [17].

Table 1. Relationship between different heuristics

	consider each result tuple independently		consider each result tuple dependently	
Intuition	cheapest	lowest average	cheapest	lowest average
heuristic	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4
function	$\|\|\{s_{\mathcal{D}}(t_j)\}\|\|_p$	$\frac{\|\|\{s_{\mathcal{D}}(t_j)\}\|\|_p}{\|w_i^{min}\|}$	$\|\|\{s_{\mathcal{D}}(t_j)\}\|\|_p$	$\frac{\|\|\{s_{\mathcal{D}}(t_j)\}\|\|_p}{\|w_i^{min} - App\_U(Q,\mathcal{D})\|}$

Heuristic 1

Algorithm 1 presents the approximation algorithm with Heuristic 1. For each result tuple t_i, the algorithm chooses the cheapest minimal why-provenance w_i^{min} (line 3). If there are more than one such minimal why-provenance, it chooses the one with the largest size (line 5). At last, it unions the chosen minimal why-provenances, and computes the price (line 6 to 7).

The complexity of choosing the cheapest minimal why-provenance of each result tuple is $O(m)$. Therefore the complexity of choosing that for all result tuples is $O(mn)$. The complexity of unioning the chosen minimal why-provenances is $O(n)$. In total, the complexity is $O(mn)$.

Algorithm 1. Heuristic 1

Data: a query Q, a database $\mathcal{D}$
Result: approximate price $App\_pr_{\mathcal{D}}(Q)$

1 **for** *each tuple* t_i *in the query result* $Q(\mathcal{D})$ **do**
2 get the set of minimal why-provenances W^i defined in [4];
3 choose the why-provenance w_i^{min} that minimizes $||\{s_{\mathcal{D}}(t_j)\}||_p$ $(t_j \in w_i^{min})$;
4 **if** *there are more the one such why-provenance* **then**
5 choose the one with largest size;
6 $App\_U(Q,\mathcal{D}) = \bigcup_i w_i^{min}$;
7 $App\_pr_{\mathcal{D}}(Q) = ||App\_U(Q,\mathcal{D})||_p$;

Heuristic 3

Algorithm 2 presents the approximation algorithm with Heuristic 3. The difference between Algorithm 2 and Algorithm 1 is that when Algorithm 2 chooses the cheapest minimal why-provenance (line 4) and compares the sizes of minimal why-provenances (line 6), it considers the source tuples that are already bought.

The complexity of choosing the cheapest minimal why-provenance of each result tuple considering the source tuples which are already bought is $O(mb)$. In total, the complexity is $O(mnb)$.

Algorithm 2. Heuristic 3

Data: a query Q, a database $\mathcal{D}$
Result: approximate price $App\_pr_{\mathcal{D}}(Q)$

while *there exist tuples not visited in the query result* $Q(\mathcal{D})$ **do**
  choose a unvisited tuple t_i uniformly at random;
  get the set of minimal why-provenances W^i defined in [4];
  choose the why-provenance w_i^{min} that minimizes $||\{s_{\mathcal{D}}(t_j)\}||_p$ $(t_j \in w_i^{min}, t_j \notin App\_U(Q,\mathcal{D}))$;
  if *there are more the one such why-provenance* **then**
    choose the one with largest size (without taking into account the ones already bought);
  $App\_U(Q,\mathcal{D}) = App\_U(Q,\mathcal{D}) \cup w_i^{min}$;
$App\_pr_{\mathcal{D}}(Q) = ||App\_U(Q,\mathcal{D})||_p$;

Example 5. The query result is $Q(\mathcal{D}) = \{t_1, t_2, t_3\}$. The sets of minimal why-provenances of t_1, t_2 and t_3 are $\{\{x_1, x_4\}, \{x_1, x_2, x_3\}\}$, $\{\{x_1, x_2\}, \{x_1, x_4\}\}$ and $\{\{x_2, x_3\}\}$, respectively. The prices of the source tuples are: $s_{\mathcal{D}}(x_1) = 1, s_{\mathcal{D}}(x_2) = 2, s_{\mathcal{D}}(x_3) = 3, s_{\mathcal{D}}(x_4) = 4$. We set $p = 1$ for p-norm.

Heuristic 1 chooses $\{x_1, x_4\}$ for t_1, since the price of $\{x_1, x_4\}$, namely 5, is lower than the price of $\{x_1, x_2, x_3\}$, namely 6. For the same reason, the algorithm chooses $\{x_1, x_2\}$ for t_2. It has to choose $\{x_2, x_3\}$ for t_3 since this is the only choice. Then the algorithm unions all the chosen minimal why-provenances, as $\{x_1, x_2, x_3, x_4\}$. Therefore the approximate price of Q is $1 + 2 + 3 + 4 = 10$.

Heuristic 3 chooses a result tuple uniformly at random each time. Let us assume that it chooses the order of t_1 first, then t_2 and at last t_3. Initially, $App\_U(Q,\mathcal{D}) = \emptyset$. Heuristic 3 chooses $\{x_1, x_4\}$ for t_1, since the price of $\{x_1, x_4\}$, namely 5, is lower than the price of $\{x_1, x_2, x_3\}$, namely 6. Now $App\_U(Q,\mathcal{D}) = \{x_1, x_4\}$. It chooses $\{x_1, x_4\}$ for t_2 since the price needed to pay for it is 0 (we already bought x_1, x_4) which is lower than the price needed to pay for $\{x_1, x_2\}$, namely 2 (we already bought x_1). Now $App\_U(Q,\mathcal{D}) = \{x_1, x_4\}$. It has to choose $\{x_2, x_3\}$ for t_3 since this is the only choice. Now $App\_U(Q,\mathcal{D}) = \{x_1, x_4, x_2, x_3\}$. Therefore the approximate price of Q is $1+2+3+4 = 10$. □

Approximability. In this section, we discuss the approximability of the above four heuristics. The authors of [10] define the *approximibility* of an approximation algorithm as its performance ratio. They show that the approximability of any approximation algorithm for this problem is always a polynomial factor to the input size. We find that the approximability of all the four approximation algorithms are $\sqrt[p]{n}$, where n is the number of result tuples and p is the p value used in Definition 11. It means that in the worst case, the approximate price could not be larger than $\sqrt[p]{n}$ times of the exact price. The worst case is as below.

Example 6. The query result $Q(\mathcal{D}) = \{t_1, t_2, t_3\}$ $(n = 3)$. The sets of minimal why-provenances of t_1, t_2, t_3 are $\{\{x_1, x_2\}, \{x_3, x_4\}\}$, $\{\{x_1, x_2\}, \{x_5, x_6\}\}$ and

$\{\{x_1, x_2\}, \{x_7, x_8\}\}$, respectively. The prices of the source tuples are: $s_{\mathcal{D}}(x_1) = s_{\mathcal{D}}(x_2) = s_{\mathcal{D}}(x_4) = s_{\mathcal{D}}(x_6) = s_{\mathcal{D}}(x_8) = 5, s_{\mathcal{D}}(x_3) = s_{\mathcal{D}}(x_5) = s_{\mathcal{D}}(x_7) = 4.99$. We set $p = 1$ for p-norm. The cheapest provenance of $Q(\mathcal{D})$ is $\{x_1, x_2\}$, hence the exact price is $pr_{\mathcal{D}}(Q) = 5+5 = 10$. However, all the 4 approximation algorithms get the same chosen provenance $\{x_3, x_4, x_5, x_6, x_7, x_8\}$, therefore the approximate price is $App\_pr_{\mathcal{D}}(Q) = 4.99 + 5 + 4.99 + 5 + 4.99 + 5 = 29.97 < 30 = 3 \times pr_{\mathcal{D}}(Q)$.

5 Empirical Performance Evaluation

In this section, we study the performance of the proposed approximation algorithms. Firstly, we study their *effectiveness*, that is how well they approximate prices. Secondly, we study their *efficiency*, that is how fast they approximate prices.

5.1 Set Up and Measurements

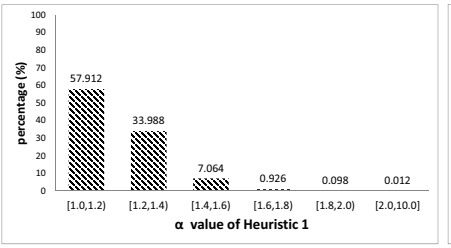
We generate a set of symbols $X = \{x_1, x_2, ..., x_b\}$, where x_i represents a source tuple in the database. We fix $b = 100$, where b is the number of source tuples that may contribute to the query result. We generate a number $s_{\mathcal{D}}(x_i)$ uniformly at random from $[1, 100]$ as the price of x_i. We generate a set of symbols $Q(\mathcal{D}) = \{t_1, t_2, ..., t_n\}$, where t_i represents a result tuple. n is fixed to be 10 for measuring the effectiveness of the approximation algorithms, while it varies from 1,000 to 5,000 for measuring the efficiency. Lastly, we generate a set of minimal why-provenances for each result tuple. We specify the maximum number of minimal why-provenances of each result tuple and the maximum number of source tuples contained in each minimal why-provenance to be 5. The numbers are generated uniformly at random from $[1, 5]$.

We measure the **effectiveness** of an approximation algorithm by computing the ratio between the approximate price and the exact price, as: $\alpha = \frac{p}{p^*}$, where p^* and p represent the price computed by the exact algorithm and the approximation algorithm, respectively. The smaller α is, the more effective this approximation algorithm is. We measure the **efficiency** of the approximation algorithms by comparing their running time. All the algorithms are implemented in Java and performed by a 2.83GHz CPU with 3.00GB RAM.

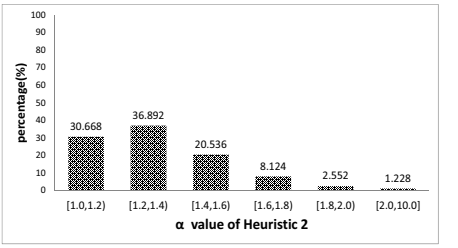
5.2 Effectiveness

50,000 random data sets are generated for evaluation, and we record α values for all the 4 approximation algorithms in each run. We set $p = 1$ for the p-norm.

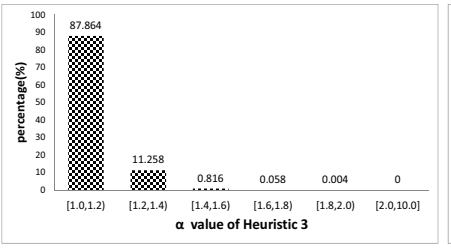
In Figure 1, we present the percentages of α values in different intervals, for individual approximation algorithms. We pre-specify some intervals for α as the x-axis of these figures, as $[1.0, 1.2), [1.2, 1.4), [1.4, 1.6), [1.6, 1.8), [1.8, 2.0), [2.0, 10.0]$. The y-axis represents the percentage of α values that fall in the corresponding interval. For instance, in Figure 1(a), the value "57.912" represents that there are 57.912% of α values in $[1.0, 1.2)$.



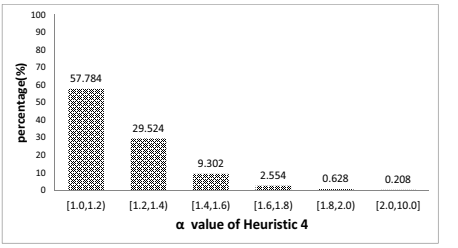
(a) Algorithm with Heuristic 1



(b) Algorithm with Heuristic 2



(c) Algorithm with Heuristic 3



(d) Algorithm with Heuristic 4

Fig. 1. Percentages of α value in different intervals

Several observations can be obtained from Figure 1. **Firstly**, for all the 4 approximation algorithms, there are rare or no α values in $[2.0, 10.0]$. Although the approximability of the approximation algorithms is 10 in our setting (according to the analysis in Section 8), it is less than 2 in most cases. **Secondly**, the algorithm with Heuristic 1 is more effective than the one with Heuristic 2, and the algorithm with Heuristic 3 is more effective than the one with Heuristic 4. From this observation, we can infer that the function used in Heuristic 1 and 3 is more suitable than the function in Heuristic 2 and 4 in the random data sets. **Thirdly**, the algorithm with Heuristic 3 is more effective than the one with Heuristic 1, and the algorithm with Heuristic 4 is more effective than the one with Heuristic 2. It can be inferred that memorizing previous choices helps to choose the current minimal why-provenance and improves the performance of the algorithms. The rationale behind this conclusion is that memorizing is helpful in making the current choice closer to the global optimal value in most cases.

5.3 Efficiency

We observe that the number of result tuples affects the efficiency most compared to other factors. Therefore in this section, we study the efficiency when we vary the number of result tuples from 1,000 to 5,000. We fix the number of minimal why-provenances for each result tuple and the number of source tuples in each minimal why-provenance to be both 5. We do not present the running time of the exact algorithm since it does not scale in this setting.

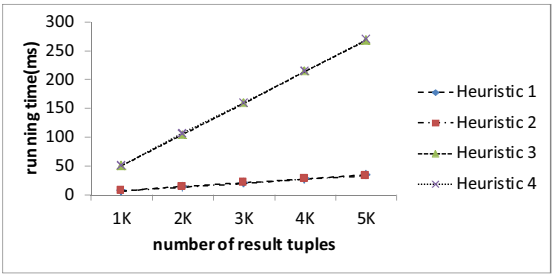


Fig. 2. Running time of different approximation algorithms

Figure 2 shows the running time of different approximation algorithms when varying the number of result tuples from 1,000 to 5,000. The x-axis represents the number of result tuples. The y-axis represents their running time. Every value is the average of 10,000 runs. Note that since the curves of Heuristic 1 and 2 are almost the same, while the curves of Heuristic 3 and 4 are almost the same, we can only see two separate curves in Figure 2.

We can get the following observations from Figure 2. **Firstly**, the algorithms with Heuristic 1 and 2 have almost the same running time, while the algorithms with Heuristic 3 and 4 also have the same running time. **Secondly**, the running time of all the algorithms is linear in the number of result tuples. **Thirdly**, the algorithms with Heuristic 1 and 2 are more efficient than the algorithms with Heuristic 3 and 4. These three observations are consistent with the theoretical complexity analysis of the approximation algorithms in Section 4.2. Generally, the approximation algorithms are efficient. For 5,000 result tuples, the running time is less than 300 ms.

6 Conclusion

In this paper we proposed a generic data pricing model that is based on minimal provenance, i.e. set of tuples contributing to the result of a query. We showed that the proposed model fulfils desirable properties such as contribution monotonicity, bounded-price and contribution arbitrage-freedom. We devised a baseline algorithm to compute the exact price of queries and also devised heuristics to approximate the price with tractable complexity. We evaluated the effectiveness and efficiency of the proposed algorithms. The empirical evaluation showed that the accuracy of the approximate price computation is much better than the theoretical analysis, and that the algorithms are efficient in practice.

Acknowledgment. This paper is partially supported by the French government under the STIC-Asia program, CCIPX project. Huayu Wu is supported by the A-STAR Grant No. 102 158 0037. Zhifeng Bao is in part supported by the Singapore NRF under its IRC@SG Funding Initiative and administered by the IDMPO, SeSame Research Center.

References

1. Andrei, S., Robert, V.: The limits of arbitrage. Journal of Finance (1997)
2. Balazinska, M., Howe, B., Suciu, D.: Data markets in the cloud: An opportunity for the database community. PVLDB 4(12), 1482–1485 (2011)
3. Brynjolfsson, E., Hitt, L.M., Kim, H.: Strength in numbers: How does data-driven decision-making affect firm performance? In: ICIS (2011)
4. Buneman, P., Khanna, S., Tan, W.-C.: Why and Where: A Characterization of Data Provenance. In: ICDT, pp. 316–330 (2001)
5. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in Databases: Why, How, and Where. Foundations and Trends in Databases 1(4), 379–474 (2009)
6. Cui, Y., Widom, J.: Practical Lineage Tracing in Data Warehouses. In: ICDE, pp. 367–378 (2000)
7. Durkee, D.: Why cloud computing will never be free. Commun. ACM (2010)
8. Hey, T., Tansley, S., Tolle, K.M. (eds.): The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research (2009)
9. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance Semirings. In: PODS (2007)
10. Khanna, S., Sudan, M., Trevisan, L., Williamson, D.P.: The approximability of constraint satisfaction problems. SIAM J. Comput. 30(6), 1863–1920 (2000)
11. Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., Suciu, D.: Query-based data pricing. In: PODS (2012)
12. Li, X.Y.: Optimization algorithms for the minimum-cost satisfiability problem
13. A. LLC. AggData, `http://www.aggdata.com/`
14. Microsoft. Windows Azure Marketplace, `https://datamarket.azure.com/`
15. Muschalle, A., Stahl, F., Loser, A., Vossen, G.: Pricing approaches for data markets. In: BIRTE (2012)
16. Prugovečki, E.: Quantum Mechanics in Hilbert Space, 2nd edn. Dover Books on Physics Series. Dover Publications (2006)
17. Tang, R., Wu, H., Bao, Z., Bressan, S., Valduriez, P.: Technical report: The price is right – a model and algorithms for the price of data, `http://www.comp.nus.edu.sg/ tang1987/tech-reports/ pricing-provenance.pdf`
18. Wu, S., Banker, R.D.: Best pricing strategy for information services. J. AIS 11(6) (2010)

What You Pay for Is What You Get

Ruiming Tang[1], Dongxu Shao[1], Stéphane Bressan[1], and Patrick Valduriez[2]

[1] National University of Singapore
{tangruiming,dcsshaod,steph}@nus.edu.sg
[2] INRIA & LIRMM, Montpellier, France
Patrick.Valduriez@inria.fr

Abstract. In most data markets, prices are prescribed and accuracy is determined by the data. Instead, we consider a model in which accuracy can be traded for discounted prices: "what you pay for is what you get".

The data market model consists of data consumers, data providers and data market owners. The data market owners are brokers between the data providers and data consumers. A data consumer proposes a price for the data that she requests. If the price is less than the price set by the data provider, then she gets an approximate value. The data market owners negotiate the pricing schemes with the data providers. They implement these schemes for the computation of the discounted approximate values.

We propose a theoretical and practical pricing framework with its algorithms for the above mechanism. In this framework, the value published is randomly determined from a probability distribution. The distribution is computed such that its distance to the actual value is commensurate to the discount. The published value comes with a guarantee on the probability to be the exact value. The probability is also commensurate to the discount. We present and formalize the principles that a healthy data market should meet for such a transaction. We define two ancillary functions and describe the algorithms that compute the approximate value from the proposed price using these functions. We prove that the functions and the algorithm meet the required principles.

1 Introduction

The three participants to a data market are data providers, data market owners and data consumers ([5]). Data providers bring data to the market and set its base price. Data consumers buy data from the market. A data market owner is a broker. She negotiates the pricing schemes with data providers and manages the market infrastructure that facilitates the transactions between data providers and data consumers.

In the pricing schemes considered in the data market literature (e.g. [8,3,7,6,2]), prices are prescribed unilaterally and data quality is the best it can be and is non-negotiable. Instead, we consider a model in which accuracy can be traded for discounted prices: "what you pay for is what you get". If a data consumer proposes to buy data at a price less than the price set by the data provider (a

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 395–409, 2013.

price less than the base price) then she can obtain an approximate value, whose quality is commensurate to the discount.

In this paper, we propose a theoretical and practical pricing framework with its algorithms for publishing an approximate value when the price is discounted. In this framework, the value published is randomly determined from a probability distribution. The distribution is computed in such a way that its distance to the actual value is commensurate to the discount sought. The published value comes with a guarantee on the probability to be the exact value. This probability also is commensurate to the discount sought.

The computation of the approximate value for the discounted price requires both a distance function and a probability function. We define an invertible distance function δ, that takes as input the price sought by the data consumer and returns the maximum distance between the degenerate distribution (i.e. the distribution that takes one value only with probability 1) for the exact value and the sampling distribution used to sample the returned value. We define an invertible probability function π, that takes as input the price offered by the data consumer and returns the minimum probability of the returned value.

The data market owner negotiates the distance and probability functions with the data providers. We formalize the principles that a healthy data market should meet for such a transaction. The distance function δ and the probability function π are inverse functions of two pricing functions, δ^{-1} and π^{-1}, respectively. δ^{-1} computes the price given the maximum distance. π^{-1} computes the price given the minimum probability. We formalize the principles of co- and contra-variance, of invariance, of a threshold requirement for these functions with respect to the price. We prove that the two functions meet the principles.

The data market owner uses the distance and probability functions to generate a probability distribution from which the published value is sampled. We argue that, in addition and in order to ensure fairness among consumers without discounting exact values, the data market owner must maximize the distance of the sampling distribution with the degenerate distribution. We devise and present algorithms to generate such an optimal distribution using the distance function δ and the probability function π.

The rest of this paper is organized as follows. Section 2 presents a rapid overview of related work on pricing. In Section 3, we present the data model and basic concepts used in our framework. Two important functions δ and π are introduced in Section 4. We study optimal satisfying distributions in Section 5, and devise algorithms to generate such an optimal distribution in Section 6. Finally, we conclude our framework in Section 7.

2 Related Work

In this section, we categorize the related works based on two dimensions. One dimension categorizes works to be data based pricing or query based pricing. Data based pricing defines prices of information goods or resources, while query based pricing defines prices of queries. The concern of the other dimension is that

whether a framework allows data consumers' willingness-to-pay (WTP) being less than the price of an item (p), i.e. $WTP < p$. Allowing $WTP < p$ means that a data consumer is free to propose any price for the data she requests.

[8,3,7,6] are in the category of data based pricing and do not allow $WTP < p$. The pricing schemes considered in [8,3] are static, which means that prices are not updated. The authors of [7,6] introduce a dynamic pricing scheme in the context of cloud computing. The prices of resources are updated when the utilization varies. However, regardless of a pricing scheme being static or dynamic, consumers have to pay the full price at the moment.

[2,4] are query based pricing. The authors of [2] propose a pricing model charging for individual queries. However, this work also only accepts full payment if a data consumer wants a query. For privacy concerns, the authors of [4] allow a partial payment. They propose a theoretic framework to assign prices to noisy query answers, as a function of a query and standard deviation. For the same query, the more accurate the answer is, the more expensive the query is. If a data consumer cannot afford the price of a query, she can choose to tolerate a higher standard deviation to lower the price.

Our framework is in the category of data based pricing and allowing $WTP < p$. The framework of [4] is similar to our work because both works return a sampled value from a generated distribution to data consumers. Our work differentiates itself from [4] in the following aspects. The framework of [4] focuses on prices of linear queries with only numerical domains, which is the same as researchers do in differential privacy; while in our framework there is a pricing function defining prices of generated distributions on any discrete domains. Moreover, in their framework, the price of a query depends on the standard derivation provided by consumers. Standard derivation is not meaningful without the exact answer of the query. In our framework, we have a probability function which states clearly that what is the probability of getting the exact value if a data consumer pays certain amount of money. It would be much easier for a data consumer to decide her payment. Even worse, in their framework a data consumer has to find an acceptable price by themselves by tuning standard deviation, which may be troublesome. In contrast, in our framework, a data consumer is able to specify the payment directly, and she gets what she pays for.

3 Data Model and Basic Concepts

In this section, we describe the data model and basic concepts of our framework. The data market model consists of data consumers, data providers and data market owners. For the ease of presentation, we consider only one data provider, one data consumer and one data market owner.

The data provider brings data to the market and sets its base price. We consider the data in relational model. A data provider owns a set of tuples. The schema is $(id, \mathbf{v})$, where id is the primary key and $\mathbf{v}$ is the value. We consider the domain of $\mathbf{v}$ to be a discrete ordered or unordered domain. All possible values of $\mathbf{v}$ are $v_1, v_2, ..., v_m$.

Although we only consider two attributes in the schema in our model, cases of more than two attributes can be handled by treating $\mathbf{v}$ as a set of attributes.

Without loss of generality, we consider the case that the data provider only owns one tuple t, $t = (\xi, v_k)$ where $k \in [1, m]$, with a base price pr_{base}. Our model can be easily extended to handle the case that the data provider owns multiple tuples, by considering each tuple independently.

The data consumer wants to buy data from the market. She requests the tuple t. However, she is not forced to pay the base price pr_{base} for it, instead, she is free to propose a price.

The data market owner is a broker between the data provider and data consumer. She negotiates the pricing scheme with the data provider. Receiving a payment from the data consumer, the data market owner returns a value, which is randomly determined from a probability distribution, to the data consumer. The distribution, namely an X-tuple [1], is generated by the data market owner according to the price a consumer is agreeable to pay.

Definition 1. *(X-tuple) An* ***X-tuple*** *is a set of m mutually exclusive tuples with the same id value but different $\boldsymbol{v}$ values, where m is the number of different possible values in the domain of $\boldsymbol{v}$. Each tuple is associated with a probability representing the confidence that this tuple is actual.*

In our framework, we consider that there is only one tuple t owned by the data provider. Under this assumption, all the X-tuples contain the same set of tuples, but with different distributions over possible values of $\mathbf{v}$, therefore, it is possible to represent an X-tuple by its distribution. As a special case, the tuple t can be represented by a degenerate distribution.

Example 1. A tuple $t = (20053046, A)$ telling that the student with *id* 20053046 gets grade A. There are four possible grades A, B, C, D. Based on t, an X-tuple $X = \{< (20053046, A), 0.3 >, < (20053046, B), 0.2 >, < (20053046, C), 0.4 >, < (20053046, D), 0.1 >\}$. X can be represented by its distribution $(0.3, 0.2, 0.4, 0.1)$. The tuple $(20053046, A)$ is sampled from X with a probability 0.3. The tuple t can be represented by a degenerate distribution $(1, 0, 0, 0)$. □

In the rest of the paper, **we represent an X-tuple by its distribution** when there is no ambiguity. **We reserve D_{base} as the degenerate distribution of the tuple t, reserve k as the index where the probability is 1 in D_{base}, reserve m as the number of tuples in an X-tuple**.

Receiving the payment pr_0 from the data consumer, the data market owner generates a distribution (i.e. an X-tuple) X satisfying two constraints:

(1) the distance between X and D_{base} is not larger than $\delta(pr_0)$;
(2) the probability of a tuple sampled from X to be t is at least $\pi(pr_0)$.

There may exist infinitely many distributions satisfying the above two constraints. In order to ensure fairness among consumers, the data market owner generates an optimal satisfying distribution (the reason is in Section 5). A distribution created is optimal in the sense that it has the maximum distance to the

degenerate distribution D_{base} under the two constraints. Finally, the data market owner samples a value from an optimal satisfying distribution and returns it to the data consumer.

4 Distance and Probability Functions

In our framework the accuracy of the value is determined from the price given by the data consumer. Therefore the distance function δ and the probability function π are inverse pricing functions. In this section we define the corresponding pricing functions, δ^{-1} and π^{-1}. δ^{-1} computes the price given the maximum distance requirement. π^{-1} computes the price given the minimum probability requirement. We formalize the principles for the exchange in terms of δ^{-1} and π^{-1} in Section 4.1 and Section 4.2, respectively.

We first recall the definition of Earth Mover's Distance, as we use it to define the pricing function δ^{-1}. To define Earth Mover's Distance, a ground distance d_{ij} is needed to measure the difference from cell i to j. We consider a family of ground distance $d_{ij} = |i - j|^q$, where $q \geq 0$. This family of ground distance is a metric, i.e. satisfying non-negative, identity of indiscernible, symmetry and triangle inequality conditions. The ground distance of $q = 0$ suits unordered domains. The ground distance of $q > 0$ suits ordered domains.

Earth mover's distance can be defined on the basis of the ground distance. Given two distributions D_1 and D_2, Earth Mover's Distance from D_1 to D_2, $EMD(D_1, D_2)$, is the optimal value of the following linear program:
$Minimize : \sum_{i,j} f_{ij} \times d_{ij} \quad s.t.$
$\forall i : \sum_j f_{ij} = D_1(i) \;\; and \;\; \forall j : \sum_i f_{ij} = D_2(j) \;\; and \;\; \forall i, j : f_{ij} \geq 0$
where $D_1(i)$ is the i^{th} cell of distribution D_1 and d_{ij} is the ground distance. Note that in our model $EMD(D_1, D_2) = EMD(D_2, D_1)$ since d_{ij} considered is symmetry. Hence, sometime in this paper, we will say that $EMD(D_1, D_2)$ is the distance **between** D_1 and D_2. In our model, EMD is easier to compute compared to general cases, because D_{base} is special. $EMD(D_{base}, X) = \sum_{i \in [1,m]} (p_i \times d_{ki})$, where p_i is the probability of the i^{th} cell of X.

4.1 The Distance Function δ and Its Inverse Function δ^{-1}

As stated in Section 3, given the payment pr_0, one of the two constraints that a generated distribution X has to satisfy is that the distance between X and D_{base} is not larger than $\delta(pr_0)$. In this section, we introduce the distance function δ, by defining and describing its inverse function δ^{-1}. δ^{-1} is a pricing function, which defines the price of a distance.

Definition 2. *(Distance function and its pricing function) A* distance function *is an invertible function $\delta : Dom_2 \to Dom_1$, where $Dom_1, Dom_2 \subseteq [0, \infty)$. It takes a price as input and returns a distance between D_{base} and the sampling distribution X.*

The inverse function of the distance function is a pricing function *$\delta^{-1} : Dom_1 \to Dom_2$, where $Dom_1, Dom_2 \subseteq [0, \infty)$. Its input is a distance (earth mover's distance) between D_{base} and X, and it returns the price.*

We have several principles for δ^{-1} that a healthy data market should meet.

Contra-variance principle: the larger the distance is, the less expensive it should be, i.e. $\Delta_1 \geq \Delta_2 \Rightarrow \delta^{-1}(\Delta_1) \leq \delta^{-1}(\Delta_2)$.

For instance, in a gold trading market, a material mixing of gold and silver has a lower price if its veracity of gold is smaller, which means that its "distance" to the pure gold is larger.

Invariance principle: each distance has only one price, i.e. $\Delta_1 = \Delta \Rightarrow \delta^{-1}(\Delta) = \delta^{-1}(\Delta_1)$.

For instance, in a gold trading market, two same materials should have the same price.

Threshold[1] principle: $\Delta \leq \Delta_U$ and $\delta^{-1}(\Delta_U) = pr_{min}$, where U is the uniform distribution, $\Delta_U = EMD(D_{base}, U)$, pr_{min} is minimum amount of money one has to pay.

We assume that X can be charged only if $EMD(D_{base}, X) \leq EMD(D_{base}, U)$. The assumption states that only when the distance between D_{base} and X is not larger than the distance between D_{base} and U, X is valuable. We make this assumption because if X is too far from D_{base}, it would be misleading to the data consumer if she buys it. In a gold trading market, there is a threshold of the veracity of gold to claim that a material is a piece of gold.

Lemma 1. *contra-variance and invariance principle imply that* $\delta^{-1}(\Delta) \leq pr_{base}$.

Based on these three principles, Dom_1 and Dom_2 in Definition 2 have to be updated accordingly: $Dom_1 \subseteq [0, \Delta_U]$ and $Dom_2 \subseteq [pr_{min}, pr_{base}]$.

We aim to design the pricing function δ^{-1} to be not only satisfying the above principles but also as smooth as possible and easy to parameterize by the data market owner. We construct the pricing function based on "sigmoid function"[2]: $S(x) = \frac{1}{1+e^{-x}}$, which is "S" shape and easily parameterized. However, applying the sigmoid function directly does not satisfy the required principles. Moreover, the domain and range of the sigmoid function do not meet our needs as well. Therefore we parameterize the sigmoid function as:

$$S_{\lambda,a}(x) = \left(\frac{1}{1+e^{-\lambda(x-a)}} - \frac{1}{1+e^{\lambda a}}\right) \times \frac{1+e^{\lambda a}}{e^{\lambda a}} \times (pr_{base} - pr_{min}) + pr_{min} \quad (1)$$

where $a > 0$ controls the inflection point and $\lambda > 0$ controls the changing rate. It is easy to verify that the range of $S_{\lambda,a}(x)$ is $[pr_{min}, pr_{base}]$ when $x \in [0, \infty)$. Since $S_{\lambda,a}(x)$ increases as x increases, it violates the contra-variance principle. Therefore we choose a kernel function: $K_1(X) = \frac{1}{EMD(D_{base},X)} - \frac{1}{EMD(D_{base},U)}$.

The pricing function δ^{-1} is a composition of K_1 and $S_{\lambda,a}$:

$$\delta^{-1}(\Delta) = \left(\frac{1}{1+e^{-\lambda(\frac{1}{\Delta}-b-a)}} - \frac{1}{1+e^{\lambda a}}\right) \times \frac{1+e^{\lambda a}}{e^{\lambda a}} \times (pr_{base} - pr_{min}) + pr_{min} \quad (2)$$

where $\Delta \in [0, \frac{1}{b}], b = \frac{1}{EMD(D_{base},U)}$, $a > 0, \lambda > 0$.

[1] The data owner is able to define another special distribution to be the threshold.

[2] Other functions are also possible. In this paper, we focus on adapting the sigmoid function.

Theorem 1. *The pricing function δ^{-1} satisfies contra-variance, invariance and threshold principles.*

Proof. Due to space limitation, we only show the sketch of the proof:

(1) $(\delta^{-1})'(\Delta) = \lambda \times (1 + \frac{1}{e^{\lambda a}}) \frac{e^{-\lambda(\frac{1}{\Delta} - b - a)}}{(1+e^{-\lambda(\frac{1}{\Delta} - b - a)})^2} \times (-\frac{1}{\Delta^2}) < 0.$

(2) It is satisfied since δ^{-1} is a function.

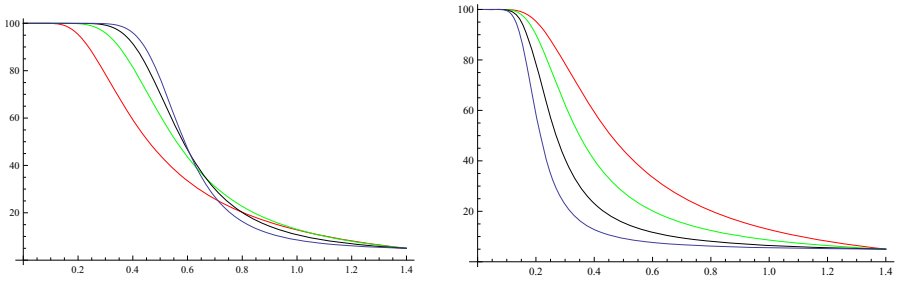
(3) $\delta^{-1}(EMD(D_{base}, U)) = 0 + pr_{min} = pr_{min}.$ □

After verifying δ^{-1} satisfying all the principles, we study how to tune the parameters in δ^{-1} according to different needs. There are two parameters in δ^{-1}: λ and a. λ controls the decreasing rate and a adjusts the inflection point.

In Figure 1, we present the curves of δ^{-1} varying λ and a, respectively. We set $pr_{base} = 100, pr_{min} = 5, D_{base} = (0, 1, 0, 0, 0)$ and $q = 1$ for the ground distance. Figure 1(a) shows four curves of δ^{-1} whose a value is fixed to be 1 and λ varies from 1 to 4. When λ is small ($\lambda = 1$), δ^{-1} starts to decrease earlier and decreases more slowly, compare to other curves when λ is larger ($\lambda = 2, 3, 4$).

After the data market owner chooses a proper λ value for herself, the parameter a in δ^{-1} is used to adjust the inflection point. Figure 1(b) shows four curves of δ^{-1} whose λ value is fixed to be 1 and a value varies from 1 to 4. As can be observed, the smaller a value is, the larger the x value of the inflection point is.

To conclude, if the data market owner prefers δ^{-1} to decrease earlier and more slowly, she chooses a smaller λ value; otherwise she chooses a larger λ value. After choosing λ, she can set a smaller a value if she prefers a larger x value of the inflection point; otherwise she chooses a larger a value.



(a) $a = 1$, varying $\lambda(\lambda = 1$ (red), $\lambda = 2$ (green), $\lambda = 3$ (black), $\lambda = 4$ (blue))

(b) $\lambda = 1$, varying $a(a = 1$ (red), $a = 2$ (green), $a = 3$ (black), $a = 4$ (blue))

Fig. 1.

4.2 The Probability Function π and Its Inverse Function π^{-1}

As stated in Section 3, given the payment pr_0, the other one of the two constraints that a generated distribution X has to satisfy is that the probability of a tuple sampled from X to be t is not less than $\pi(pr_0)$. In this section, we introduce the probability function π by presenting its inverse function π^{-1}. π^{-1} is a pricing function, which defines the price of a probability requirement.

Definition 3. *(Probability function and its pricing function) A* probability function *is an invertible function* $\pi : Dom_2 \rightarrow Dom_1$*, where* $Dom_1 \subseteq [0,1]$ *and* $Dom_2 \subseteq [0,\infty)$*. It takes a price as input and returns a probability of a tuple sampled from the sampling distribution* X *to be* t*.*

The inverse function of the probability function is a pricing function $\pi^{-1} : Dom_1 \rightarrow Dom_2$*, where* $Dom_1 \subseteq [0,1]$ *and* $Dom_2 \subseteq [0,\infty)$*. Its input is a probability requirement, and it returns the price.*

We have three principles for the pricing function π^{-1}, which are similar to the ones for δ^{-1}.
Co-variance principle: the larger the probability is, the more expensive it should be, i.e. $p_1 \geq p_2 \Rightarrow \pi^{-1}(p_1) \geq \pi^{-1}(p_2)$.
Invariance principle: for any p, its has only one price, i.e. $p_1 = p \Rightarrow \pi^{-1}(p) = \pi^{-1}(p_1)$.
Threshold[3] principle: $p \geq \frac{1}{m}$ and $\pi^{-1}(\frac{1}{m}) = pr_{min}$.

Lemma 2. *co-variance and invariance principles imply that* $\pi^{-1}(p) \leq pr_{base}$*.*

Based on these three principles, Dom_1 and Dom_2 in Definition 3 have to be updated accordingly: $Dom_1 \subseteq [\frac{1}{m}, 1]$ and $Dom_2 \subseteq [pr_{min}, pr_{base}]$.

Using the same strategy as developing δ^{-1}, we choose a kernel function: $K_2(p) = \frac{1}{1-p} - \frac{m}{m-1}$.

The pricing function π^{-1} is a composition of $S_{\lambda,a}$ (Equation 1) and K_2:

$$\pi^{-1}(p) = \left(\frac{1}{1+e^{-\lambda(\frac{1}{1-p}-\frac{m}{m-1}-a)}} - \frac{1}{1+e^{\lambda a}}\right) \times \frac{1+e^{\lambda a}}{e^{\lambda a}} \times (pr_{base} - pr_{min}) + pr_{min} \tag{3}$$

where λ and a have **the same value** as the corresponding value in δ^{-1}. Note that π^{-1} can be derived from δ^{-1} by applying the ground distance of $q = 1$.

Theorem 2. *The pricing function* π^{-1} *satisfies co-variance, invariance and threshold principles.*

Proof. We only present the sketch of the proof due to space limitation.
(1) $(\pi^{-1})'(p) = \lambda \times (1 + \frac{1}{e^{\lambda a}}) \frac{e^{-\lambda(\frac{1}{1-p}-b-a)}}{(1+e^{-\lambda(\frac{1}{1-p}-b-a)})^2} \times (-\frac{1}{(1-p)^2}) \times (-1) > 0$.
(2) It is satisfied since π^{-1} is a function.
(3) $\pi^{-1}(\frac{1}{m}) = 0 + pr_{min} = pr_{min}$. □

5 Optimal $(pr_0, D_{base})-$Acceptable Distributions

In this section, we study optimal distributions that satisfy the two constraints given by the distance function δ and the probability function π. Recall that given the payment pr_0 by the data consumer, the data market owner generates a distribution X (on the basis of t) satisfying two constraints:

[3] The data owner is able to define a special probability (other than $\frac{1}{m}$) to be the threshold.

(1) $EMD(D_{base}, X) \leq \delta(pr_0)$;
(2) $p_k \geq \pi(pr_0)$, where k is the dimension that $v_k = t.v$.

We define such satisfying distributions to be $(pr_0, D_{base})-$**acceptable**. D_{base} is a $(pr_0, D_{base})-$acceptable distribution. However, the data market owner should try to avoid generating D_{base}, otherwise no consumer will pay pr_{base} for it.

Definition 4. *(Optimal distribution) A $(pr_0, D_{base})-$acceptable distribution X_{opt} is* optimal *if $EMD(D_{base}, X_{opt}) \geq EMD(D_{base}, Y)$, where Y is an arbitrary $(pr_0, D_{base})-$acceptable distribution.*

There may exist infinitely many $(pr_0, D_{base})-$acceptable distributions. In order to ensure fairness among consumers, the data market owner generates an optimal $(pr_0, D_{base})-$acceptable distribution. If she arbitrarily generates a $(pr_0, D_{base})-$acceptable distribution, some unfair cases may happen. It is possible that when a consumer c_1 pays more than a second consumer c_2, the distribution X_1 for c_1 is less expensive than the distribution X_2 for c_2. This is unfair for c_1. We will show that generating an optimal $(pr_0, D_{base})-$acceptable distribution prevents such kind of unfairness.

In the rest of this section, we aim to find $EMD(D_{base}, X_{opt})$ where X_{opt} is an optimal $(pr_0, D_{base})-$acceptable distribution. For the ease of presentation, given pr_0, m and D_{base}, let $s = S^{-1}_{\lambda,a}(pr_0)$, $\alpha = \frac{m}{m-1}$ and $\beta = \frac{1}{EMD(D_{base},U)}$. **These symbols will be used in the theoretic analysis and in the algorithms.** From Equations 1, 2 and 3, we have

$\frac{1}{s+\beta} = (\delta^{-1})^{-1}(pr_0) = \delta(pr_0)$ *and* $1 - \frac{1}{s+\alpha} = (\pi^{-1})^{-1}(pr_0) = \pi(pr_0)$

Lemma 3. *When $m = 2$ and $k = 1$, the optimal $(pr_0, D_{base})-$acceptable distribution is $(1 - \frac{1}{s+\beta}, \frac{1}{s+\beta})$.*

Proof. When $m = 2$ and $k = 1$, $D_{base} = (1, 0), U = (\frac{1}{2}, \frac{1}{2}), X = (p_1^X, p_2^X)$[4]. Therefore $EMD(D_{base}, X) = p_2^X$ and $\alpha = \beta = 2$.

A distribution X is $(pr_0, D_{base})-$acceptable if and only if

$$\begin{cases} EMD(D_{base}, X) \leq \delta(pr_0) \Leftrightarrow p_2^X \leq \delta(pr_0) = \frac{1}{s+\beta} \\ p_1^X \geq \pi(pr_0) \Leftrightarrow p_2^X \leq 1 - \pi(pr_0) = \frac{1}{s+\alpha} \end{cases}$$

$\alpha = \beta$, therefore the maximum $EMD(D_{base}, X) = \frac{1}{s+\beta}$ is obtained by setting $p_2^X = \frac{1}{s+\beta}$ and $p_1^X = 1 - \frac{1}{s+\beta}$. □

Corollary 1. *When $m = 2$ and $k = 2$, the optimal $(pr_0, D_{base})-$acceptable distribution is $(\frac{1}{s+\beta}, 1 - \frac{1}{s+\beta})$.*

Corollary 2. *When $m = 3$ and $k = 2$, an optimal $(pr_0, D_{base})-$acceptable distribution is $(\epsilon, 1 - \frac{1}{s+\beta}, \frac{1}{s+\beta} - \epsilon)$, where $\epsilon \in [0, \frac{1}{s+\beta}]$.*

Lemma 4. *When $m = 3$ and $k = 1$, the distance between D_{base} and an optimal $(pr_0, D_{base})-$acceptable distribution is* $\min\{\frac{1}{s+\beta}, \frac{2^q}{s+\alpha}\}$.

[4] In this paper, we use p_l^X to represent the probability in the l^{th} cell of distribution X.

Proof. When $m = 3$ and $k = 1$, $EMD(D_{base}, X) = p_2^X + 2^q p_3^X$.

A distribution X is $(pr_0, D_{base})-$acceptable if and only if
$\begin{cases} EMD(D_{base}, X) \leq \delta(pr_0) \Leftrightarrow p_2^X + 2^q p_3^X \leq \delta(pr_0) = \frac{1}{s+\beta} \\ p_1^X \geq \pi(pr_0) \Leftrightarrow p_2^X + p_3^X \leq 1 - \pi(pr_0) = \frac{1}{s+\alpha} \end{cases}$

$EMD(D_{base}, X)$ reaches its maximum $\frac{1}{s+\beta}$ by setting $p_2^X = \frac{1}{2^q-1}(\frac{2^q}{s+\alpha} - \frac{1}{s+\beta})$ and $p_3^X = \frac{1}{2^q-1}(\frac{1}{s+\beta} - \frac{1}{s+\alpha})$, in which case the above two equalities hold.

There is another requirement which is $p_2^X \geq 0, p_3^X \geq 0$. p_3^X is always non-negative since $\alpha \geq \beta$. However, p_2^X will be negative if $\frac{1}{s+\beta} > \frac{2^q}{s+\alpha}$.

- If $\frac{1}{s+\beta} \leq \frac{2^q}{s+\alpha}$, $p_2^X = \frac{1}{2^q-1}(\frac{2^q}{s+\alpha} - \frac{1}{s+\beta}) \geq 0$ and $p_3^X = \frac{1}{2^q-1}(\frac{1}{s+\beta} - \frac{1}{s+\alpha}) \geq 0$. In this case, the maximum $EMD(D_{base}, X)$ is $\frac{1}{s+\beta}$.
- If $\frac{1}{s+\beta} > \frac{2^q}{s+\alpha}$, the maximum $EMD(D_{base}, X)$ is obtained by setting $p_2^X = 0$ and $p_3^X = \frac{1}{s+\alpha}$. $EMD(D_{base}, X) = \frac{2^q}{s+\alpha}$.

The maximum $EMD(D_{base}, X)$ is $\min\{\frac{1}{s+\beta}, \frac{2^q}{s+\alpha}\}$. Therefore the distance between D_{base} and an optimal $(pr_0, D_{base})-$acceptable distribution is $\min\{\frac{1}{s+\beta}, \frac{2^q}{s+\alpha}\}$.

The proof of Lemma 4 can be generalized to cases of $m > 3$. Before going to the generalization, we first prove a lemma which gives a uniform representation for the distributions with the same distance from D_{base}.

Definition 5. *(D_{base} equivalence) X, Y are two distributions. We say $X \equiv_{D_{base}} Y$ if $EMD(D_{base}, X) = EMD(D_{base}, Y)$ and $p_k^X = p_k^Y$.*

Definition 6. *($D_{base}-$form) Let $m > 3$ and X be a distribution. X is in $D_{base}-$**form** if $p_l^X = 0$ for $l \neq k, i, j$, where i is an index nearest to k and j is an index furthest from k. If there exist more than one such i, j, we randomly choose one i and j. For $l = k, i, j, p_l^X = 0$ or $p_l^X \neq 0$.*

Example 2. Assume $D_{base} = (0, 1, 0, 0, 0)$. k is the index where the probability is 1 in D_{base}, i.e. $k = 2$; j is an index furthest from k, i.e. $j = 5$; i is an index nearest to k, i.e. $i = 1$ or $i = 3$. We randomly choose $i = 1$. Therefore, a possible distribution in $D_{base}-$form is $X_1 = (0.2, 0.7, 0, 0, 0.1)$. □

Lemma 5. *Let $m > 3$ and X be an arbitrary distribution. There is always a distribution Y in $D_{base}-$form such that $X \equiv_{D_{base}} Y$.*

Proof. Fix i, j and k as in Definition 6. For each $l \neq i, j, k$ such that $p_l^X \neq 0$, we allocate p_l^X to p_i^Y and p_j^Y. Let $A_{l,i}$ be the flow from p_l^X to p_i^Y. Let $A_{l,j}$ be the flow from p_l^X to p_j^Y. To preserve $X \equiv_{D_{base}} Y$, the following equations hold:

$$\begin{cases} A_{l,i} + |j-k|^q A_{l,j} = |l-k|^q p_l^X \\ A_{l,i} + A_{l,j} = p_l^X \end{cases}$$

Hence we have $A_{l,i} = \frac{|j-k|^q - |l-k|^q}{|j-k|^q - 1} p_l^X$, $A_{l,j} = \frac{|l-k|^q - 1}{|j-k|^q - 1} p_l^X$.

j being furthest from k infers that $|j-k|^q - |l-k|^q \geq 0$. Since $m > 3$, $|j-k|^q - 1$ is always positive. Therefore, $A_{l,i}$ and $A_{l,j}$ are always non-negative.

To construct Y, let $p_k^Y = p_k^X$, $p_i^Y = p_i^X + \sum_{l \neq i,j,k} A_{l,i}$, $p_j^Y = p_j^X + \sum_{l \neq i,j,k} A_{l,j}$ and $p_l^Y = 0$ for $l \neq i, j, k$. □

Extending Lemma 4, the theorem below presents the distance between D_{base} and an optimal (pr_0, D_{base})-acceptable distribution, in the general case of $m > 3$.

Theorem 3. *Let $m > 3$ and X_{opt} be an optimal (pr_0, D_{base})-acceptable distribution. $EMD(D_{base}, X_{opt}) = \min\{\frac{|j-k|^q}{s+\alpha}, \frac{1}{s+\beta}\}$.*

Proof. Due to Lemma 5, we only have to consider X in $D_{base}-$form. Recall that the probability of the k^{th} cell of D_{base} is 1. Fix an index i such that $|i-k| = 1$ and fix an index j maximizing $|j-k|$. $p_l^X = 0$ for $l \neq k, i, j$.

X is (pr_0, D_{base})-acceptable if and only if:

$$\begin{cases} EMD(D_{base}, X) \leq \delta(pr_0) \Leftrightarrow p_i^X + |j-k|^q p_j^X \leq \frac{1}{s+\beta} \\ p_k^X \geq \pi(pr_0) \Leftrightarrow p_i^X + p_j^X \leq \frac{1}{s+\alpha} \end{cases}$$

– If $\frac{|j-k|^q}{s+\alpha} \geq \frac{1}{s+\beta}$, by solving the equations below:

$$\begin{cases} p_i^X + |j-k|^q p_j^X = \frac{1}{s+\beta} \\ p_i^X + p_j^X = \frac{1}{s+\alpha} \end{cases}$$

we have $p_i^X = \frac{1}{|j-k|^q - 1}(\frac{|j-k|^q}{s+\alpha} - \frac{1}{s+\beta}), p_j^X = \frac{1}{|j-k|^q - 1}(\frac{1}{s+\beta} - \frac{1}{s+\alpha})$.
p_j^X is always non-negative since $\frac{1}{s+\beta} \geq \frac{1}{s+\alpha}$ (the same reason as in Lemma 4).
p_i^X is also non-negative since $\frac{|j-k|^q}{s+\alpha} \geq \frac{1}{s+\beta}$. In this setting $EMD(D_{base}, X) = \frac{1}{s+\beta}$, which is the maximum value that $EMD(D_{base}, X)$ can reach.

– If $\frac{|j-k|^q}{s+\alpha} < \frac{1}{s+\beta}$, let $p_i^X = 0$ and $p_j^X = \frac{1}{s+\alpha}$. Then X is $(pr_0, D_{base})-$acceptable and $EMD(D_{base}, X) = \frac{|j-k|^q}{s+\alpha}$. Below we prove that this value is the maximum value that $EMD(D_{base}, X)$ can reach in this case. Let Y be an arbitrary $(pr_0, D_{base})-$acceptable distribution in $D_{base}-$form. Then $p_i^Y + |j-k|^q p_j^Y \leq \frac{1}{s+\beta}$ and $p_i^Y + p_j^Y \leq \frac{1}{s+\alpha}$. Hence

$$\begin{aligned} &EMD(D_{base}, Y) = p_i^Y + |j-k|^q p_j^Y = p_i^Y + p_j^Y + (|j-k|^q - 1)p_j^Y \\ &\leq \frac{1}{s+\alpha} + (|j-k|^q - 1)\frac{1}{s+\alpha} = \frac{|j-k|^q}{s+\alpha} = EMD(D_{base}, X) \end{aligned}$$

When $p_i^Y = 0$ and $p_j^Y = \frac{1}{s+\alpha}$, the equality holds.

To conclude, the maximum value of $EMD(D_{base}, X)$ is $\min\{\frac{|j-k|^q}{s+\alpha}, \frac{1}{s+\beta}\}$, for $m > 3$. Therefore $EMD(D_{base}, X_{opt}) = \min\{\frac{|j-k|^q}{s+\alpha}, \frac{1}{s+\beta}\}$. □

Remark 1. Combining the results of Corollary 1, 2, Lemma 3, 4 and Theorem 3, we can conclude that when $m > 1$, $EMD(D_{base}, X_{opt}) = \min\{\frac{|j-k|^q}{s+\alpha}, \frac{1}{s+\beta}\}$. □

Generating an optimal $(pr_0, D_{base})-$acceptable distribution ensures fairness among consumers, as we have stated in the beginning of this section. We prove it by the lemma below.

Lemma 6. *Assume consumer c_1, c_2 pay pr_1, pr_2 respectively and the data market owner generates optimal $(pr_1, D_{base})-$acceptable and $(pr_2, D_{base})-$acceptable X-tuples X_1, X_2 respectively. It is true that $pr_1 \geq pr_2 \Rightarrow \delta^{-1}(EMD(D_{base}, X_1)) \geq \delta^{-1}(EMD(D_{base}, X_2))$.*

Proof. Assume $pr_1 \geq pr_2$. Due to the contra-variance principle of δ^{-1}, $\frac{1}{s_1+\beta} = \delta(pr_1) \leq \frac{1}{s_2+\beta} = \delta(pr_2)$. Due to the co-variance principle of π^{-1}, $1 - \frac{1}{s_1+\alpha} = \pi(pr_1) \geq 1 - \frac{1}{s_2+\alpha} = \pi(pr_2)$, which means $\frac{1}{s_1+\alpha} \leq \frac{1}{s_2+\alpha}$. X_1, X_2 are optimal distributions for pr_1, pr_2 respectively. Therefore, from Theorem 3, we have $EMD(D_{base}, X_1) = \min\{\frac{|j-k|^q}{s_1+\alpha}, \frac{1}{s_1+\beta}\} \leq EMD(D_{base}, X_2) = \min\{\frac{|j-k|^q}{s_2+\alpha}, \frac{1}{s_2+\beta}\}$.

Therefore $\delta^{-1}(EMD(D_{base}, X_1)) \geq \delta^{-1}(EMD(D_{base}, X_2))$. □

6 Algorithms

We present algorithms to generate an optimal $(pr_0, D_{base})-$acceptable distribution in this section. Firstly we present an algorithm which generates an optimal $(pr_0, D_{base})-$acceptable distribution in D_{base}-form. However, D_{base}-form may not always meet the needs. Hence we present another algorithm transforming an optimal $(pr_0, D_{base})-$acceptable distribution X_{opt} in $D_{base}-$form to another optimal distribution Y_{opt} which is not in D_{base}-form, while preserving $X_{opt} \equiv_{D_{base}} Y_{opt}$.

In this section, we consider the case of the domain size $m > 3$. The complete discussion on the cases of $m = 2, 3$ are omitted due to space limit. When $m = 2$, an optimal $(pr_0, D_{base})-$acceptable distribution can be generated according to Lemma 3 and Corollary 1; while an optimal $(pr_0, D_{base})-$acceptable distribution of $m = 3$ can be produced according to Corollary 2 and Lemma 4.

We use X_{opt} to represent an optimal $(pr_0, D_{base})-$acceptable distribution. Algorithm 1 generates X_{opt} in $D_{base}-$form. It is devised based on the proof of Theorem 3.

However, the data market owner may not want D_{base}-form in some cases. Therefore, we devise an algorithm to transform X_{opt} in $D_{base}-$form to another optimal distribution Y_{opt} in non-D_{base}-form while preserving $X_{opt} \equiv_{D_{base}} Y_{opt}$. Below we discuss the transformation in different cases individually and then summarize them in an algorithm.

Firstly, We Consider the Case of $\frac{|j-k|^q}{s+\alpha} \geq \frac{1}{s+\beta}$. In this case, we observe from Algorithm 1 that $p_i^{X_{opt}}, p_j^{X_{opt}}, p_k^{X_{opt}} \neq 0$. To preserve $X_{opt} \equiv_{D_{base}} Y_{opt}$, we have to make sure that $p_k^{X_{opt}} = p_k^{Y_{opt}}$ and $EMD(D_{base}, X_{opt}) = EMD(D_{base}, Y_{opt})$. We consider the following sub-cases.

Algorithm 1. Generating X_{opt} in $D_{base}-$form

Data: $D_{base}, pr_0, \pi, \delta$
Result: X_{opt} in $D_{base}-$form

if $\frac{|j-k|^q}{s+\alpha} \geq \frac{1}{s+\beta}$ **then**
  $p_i^{X_{opt}} = \frac{1}{|j-k|^q-1}(\frac{|j-k|^q}{s+\alpha} - \frac{1}{s+\beta})$; $p_j^{X_{opt}} = \frac{1}{|j-k|^q-1}(\frac{1}{s+\beta} - \frac{1}{s+\alpha})$;
  $p_k^{X_{opt}} = 1 - \frac{1}{s+\alpha}$;
else
  $p_i^{X_{opt}} = 0$; $p_j^{X_{opt}} = \frac{1}{s+\alpha}$; $p_k^{X_{opt}} = 1 - \frac{1}{s+\alpha}$;

Subcase 1: $p_i^{X_{opt}} < p_i^{Y_{opt}}, p_j^{X_{opt}} < p_j^{Y_{opt}}$. It is impossible to transform to Y_{opt}. Since $p_k^{X_{opt}} = p_k^{Y_{opt}}$ and all the probabilities other than cells i, j, k are 0, no flow can go to $p_i^{Y_{opt}}$ and $p_j^{Y_{opt}}$.

Subcase 2: $p_i^{X_{opt}} < p_i^{Y_{opt}}, p_j^{X_{opt}} > p_j^{Y_{opt}}$. Assume $p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1$ ($\epsilon_1 \in (0, p_j^{X_{opt}})$), and $p_i^{Y_{opt}} = p_i^{X_{opt}} + \epsilon_2$ ($\epsilon_2 \in (0, \epsilon_1)$). Without loss of generality, we assume that there is another cell $p_a^{Y_{opt}} = \epsilon_1 - \epsilon_2$. The probabilities of Y_{opt} are: $p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1, p_i^{Y_{opt}} = p_i^{X_{opt}} + \epsilon_2, p_a^{Y_{opt}} = \epsilon_1 - \epsilon_2, p_k^{Y_{opt}} = p_k^{X_{opt}}$.

$$
\begin{aligned}
&EMD(D_{base}, Y_{opt}) \\
&= (p_j^{X_{opt}} - \epsilon_1)|j-k|^q + (p_i^{X_{opt}} + \epsilon_2)|i-k|^q + (\epsilon_1 - \epsilon_2)|a-k|^q \\
&= p_j^{X_{opt}}|j-k|^q + p_i^{X_{opt}}|i-k|^q + \epsilon_2(|i-k|^q - |j-k|^q) + (\epsilon_1 - \epsilon_2)(|a-k|^q - |j-k|^q) \\
&< p_j^{X_{opt}}|j-k|^q + p_i^{X_{opt}}|i-k|^q = EMD(D_{base}, X_{opt})
\end{aligned}
$$

The reason for "<" is that $|j-k| > |i-k|$ and $|j-k| > |a-k|$. Therefore, this case is impossible for the transformation.

Subcase 3: $p_i^{X_{opt}} > p_i^{Y_{opt}}, p_j^{X_{opt}} < p_j^{Y_{opt}}$. This case is also impossible for the transformation, since $EMD(D_{base}, Y_{opt}) > EMD(D_{base}, X_{opt})$. The reason is similar to that of the previous case, therefore we omit it due to space limit.

Subcase 4: when $p_i^{X_{opt}} = p_i^{Y_{opt}}$, and there exists an index a such that $|a-k| = |j-k|$, we can have $EMD(D_{base}, X_{opt}) = EMD(D_{base}, Y_{opt})$ if $p_a^{Y_{opt}} = \epsilon$ and $p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon$ ($\epsilon \in (0, p_j^{X_{opt}})$). Similarly, when $p_j^{X_{opt}} = p_j^{Y_{opt}}$, and there exists an index a such that $|a-k| = |i-k|$, we can have $EMD(D_{base}, X_{opt}) = EMD(D_{base}, Y_{opt})$ if $p_a^{Y_{opt}} = \epsilon$ and $p_i^{Y_{opt}} = p_i^{X_{opt}} - \epsilon$ ($\epsilon \in (0, p_i^{X_{opt}})$).

Subcase 5: $p_i^{X_{opt}} > p_i^{Y_{opt}}, p_j^{X_{opt}} > p_j^{Y_{opt}}$. Assume $p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1$ ($\epsilon_1 \in (0, p_j^{X_{opt}}]$), and $p_i^{Y_{opt}} = p_i^{X_{opt}} - \epsilon_2$ ($\epsilon_2 \in (0, p_i^{X_{opt}}]$). We consider that the flow $\epsilon_1 + \epsilon_2$ is transferred to **only one cell**, namely the cell a. This cell is selected **uniformly at random** from the cells other than i, j and k. Moreover, we have to avoid $|a-k| = |i-k|$ and $|a-k| = |j-k|$. The probabilities of Y_{opt} are:
$p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1, p_i^{Y_{opt}} = p_i^{X_{opt}} - \epsilon_2, p_a^{Y_{opt}} = \epsilon_1 + \epsilon_2, p_k^{Y_{opt}} = p_k^{X_{opt}}$.

By solving the equation $EMD(D_{base}, X_{opt}) = EMD(D_{base}, Y_{opt})$, we have $\frac{\epsilon_1}{\epsilon_2} = \frac{|a-k|^q-|i-k|^q}{|j-k|^q-|a-k|^q}$. $\epsilon_2 = \frac{|j-k|^q-|a-k|^q}{|a-k|^q-|i-k|^q}\epsilon_1 \leq p_i^{X_{opt}}$, therefore $\epsilon_1 \leq \frac{|a-k|^q-|i-k|^q}{|j-k|^q-|a-k|^q}p_i^{X_{opt}}$. Let $L = \min\{p_j^{X_{opt}}, \frac{|a-k|^q-|i-k|^q}{|j-k|^q-|a-k|^q}p_i^{X_{opt}}\}$. We sample ϵ_1 from $(0, L]$ uniformly at random. Then ϵ_2 can be computed using the above equation. Once ϵ_1 and ϵ_2 is computed, Y_{opt} is produced.

Secondly, We Consider the Case of $\frac{|j-k|^q}{s+\alpha} < \frac{1}{s+\beta}$. In this case, we observe from Algorithm 1 that only $p_j^{X_{opt}}, p_k^{X_{opt}} \neq 0$. In order to keep $X_{opt} \equiv_{D_{base}} Y_{opt}$, we have to make sure that $p_k^{X_{opt}} = p_k^{Y_{opt}}$ and $EMD(D_{base}, X_{opt}) = EMD(D_{base}, Y_{opt})$. We consider the following sub-cases.

Subcase 1: $p_j^{X_{opt}} < p_j^{Y_{opt}}$. It is impossible to transform to Y_{opt}. Since $p_k^{X_{opt}} = p_k^{Y_{opt}}$ and all the probabilities other than cells j, k are 0, no flow can go to $p_j^{Y_{opt}}$.

Subcase 2: $p_j^{X_{opt}} > p_j^{Y_{opt}}$. Assume $p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1$ ($\epsilon_1 \in (0, p_j^{X_{opt}})$), and without loss of generality, we assume ϵ_1 is transferred to $p_a^{Y_{opt}}$. Therefore, the probabilities of Y_{opt} are: $p_i^{Y_{opt}} = 0, p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1, p_a^{Y_{opt}} = \epsilon_1, p_k^{Y_{opt}} = p_k^{X_{opt}}$.

Algorithm 2. Transforming X_{opt} to non-D_{base}-form Y_{opt}

Data: X_{opt} in $D_{base}-$form
Result: a non-D_{base}-form Y_{opt} such that $X_{opt} \equiv_{D_{base}} Y_{opt}$

if $\frac{|j-k|^q}{s+\alpha} < \frac{1}{s+\beta}$ **then**
  if *there exists an index a that* $|j-k| = |a-k|$ **then**
    sample ϵ_1 from $(0, p_j^{X_{opt}})$;
    the probabilities of Y_{opt} are $p_i^{Y_{opt}} = 0, p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1, p_a^{Y_{opt}} = \epsilon_1, p_k^{Y_{opt}} = p_k^{X_{opt}}$;
else if $\frac{|j-k|^q}{s+\alpha} \geq \frac{1}{s+\beta}$ **then**
  sample an index a from cells other than i, j, k;
  if $|a-k| = |j-k|$ **then**
    sample ϵ_1 from $(0, p_j^{X_{opt}})$;
    the probabilities of Y_{opt} are $p_i^{Y_{opt}} = p_i^{X_{opt}}, p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1, p_a^{Y_{opt}} = \epsilon_1, p_k^{Y_{opt}} = p_k^{X_{opt}}$;
  else if $|a-k| = |i-k|$ **then**
    sample ϵ_1 from $(0, p_i^{X_{opt}})$;
    the probabilities of Y_{opt} are $p_i^{Y_{opt}} = p_i^{X_{opt}} - \epsilon_1, p_j^{Y_{opt}} = p_j^{X_{opt}}, p_a^{Y_{opt}} = \epsilon_1, p_k^{Y_{opt}} = p_k^{X_{opt}}$;
  else
    sample ϵ_1 from $(0, \min\{p_j^{X_{opt}}, \frac{|a-k|^q-|i-k|^q}{|j-k|^q-|a-k|^q}p_i^X\}]$;
    $\epsilon_2 = \frac{|j-k|^q-|a-k|^q}{|a-k|^q-|i-k|^q}\epsilon_1$;
    the probabilities of Y_{opt} are $p_i^{Y_{opt}} = p_i^{X_{opt}} - \epsilon_2, p_j^{Y_{opt}} = p_j^{X_{opt}} - \epsilon_1, p_a^{Y_{opt}} = \epsilon_1 + \epsilon_2, p_k^{Y_{opt}} = p_k^{X_{opt}}$;

Only when $|j-k| = |a-k|$, $EMD(D_{base}, X_{opt}) = EMD(D_{base}, Y_{opt})$.

Based on the above discussion, we devise Algorithm 2 transforming an optimal $(pr_0, D_{base})-$acceptable distribution X_{opt} in $D_{base}-$form to a non-D_{base}-form distribution Y_{opt} while preserving $Y_{opt} \equiv_{D_{base}} X_{opt}$.

Finally, the data market owner samples a value from a distribution generated by the algorithms, and returns it to the data consumer.

7 Conclusion

In this paper, we proposed a theoretical and practical pricing framework for a data market in which data consumers can trade data quality for discounted prices. In our framework, the value provided to a data consumer is exact if she offers the full price for it. The value is approximate if she offers to pay only a discounted price. In the case of a discounted price, the value is randomly determined from a probability distribution. The distance of the distribution to the actual value (to the degenerate distribution) is commensurate to the discount. The published value comes with a guarantee on its probability. The probability is also commensurate to the discount. We defined ancillary pricing functions and the necessary algorithms under several principles for a healthy market. We proved the correctness of the functions and algorithms.

Acknowledgment. This paper is partially supported by the French government under the STIC-Asia program, CCIPX project. Dongxu Shao is funded by the A*Star SERC project "Hippocratic Data Stream Cloud for Secure, Privacy-preserving Data Analytics Services" 102 158 0037, NUS Ref: R-252-000-433-305.

References

1. Benjelloun, O., Sarma, A.D., Halevy, A.Y., Widom, J.: Uldbs: Databases with uncertainty and lineage. In: VLDB, pp. 953–964 (2006)
2. Koutris, P., Upadhyaya, P., Balazinska, M., Howe, B., Suciu, D.: Query-based data pricing. In: PODS (2012)
3. Kushal, A., Moorthy, S., Kumar, V.: Pricing for data markets. Technical Report (2012)
4. Li, C., Li, D.Y., Miklau, G., Suciu, D.: A theory of pricing private data. In: ICDT, pp. 33–44 (2013)
5. Muschalle, A., Stahl, F., Loser, A., Vossen, G.: Pricing approaches for data markets. In: BIRTE (2012)
6. Püschel, T., Anandasivam, A., Buschek, S., Neumann, D.: Making money with clouds: Revenue optimization through automated policy decisions. In: ECIS, pp. 2303–2314 (2009)
7. Püschel, T., Neumann, D.: Management of cloud infastructures: Policy-based revenue optimization. In: ICIS, p. 178 (2009)
8. Wu, S., Banker, R.D.: Best pricing strategy for information services. J. AIS 11(6) (2010)

A File Recommendation Method Based on Task Workflow Patterns Using File-Access Logs

Qiang Song[1], Takayuki Kawabata[2], Fumiaki Itoh[2],
Yousuke Watanabe[1], and Haruo Yokota[1]

[1] Tokyo Institute of Technology
[2] Canon Inc.
{soukyou,watanabe}@de.cs.titech.ac.jp,
{kawabata.takayuki,ito.fumiaki}@canon.co.jp, yokota@cs.titech.ac.jp

Abstract. In recent years, office workers spend much time and effort searching for the documents required for their jobs. To reduce these costs, we propose a new method for recommending files and operations on them. Existing technologies for recommendation, such as collaborative filtering, suffer from two problems. First, they can only work with documents that have been accessed in the past, so that they cannot recommend when only newly generated documents are inputted. Second, they cannot easily handle sequences involving similar or differently ordered elements because of the strict matching used in the access sequences. To solve these problems, such minor variations should be ignored. In our proposed method, we introduce the concepts of abstract files as groups of similar files used for a similar purpose, abstract tasks as groups of similar tasks, and frequent abstract workflows grouped from similar workflows, which are sequences of abstract tasks. In experiments using real file-access logs, we confirmed that our proposed method could extract workflow patterns with longer sequences and higher support-count values, which are more suitable as recommendations.

Keywords: File recommendation, File abstraction, Abstract task, Abstract workflow, Log analysis.

1 Introduction

The numbers of files in file systems have grown dramatically. As a consequence of this increase, office workers now spend much time and effort searching for files that include documents and data required for their business [1]. It is therefore important to find methods for reducing the time wasted in ineffective and unproductive searching. Also, in office, the searching targets are not limited to documents, the abstract working processes which we call them abstract workflows in this paper are also being searched by users.

Collaborative filtering is well known as an algorithm for making recommendations. However, it only works with files accessed in the past. Recommendation systems based on this algorithm do not work well on newly generated files. Moreover, collaborative filtering does not handle well sequences that include elements

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 410–417, 2013.

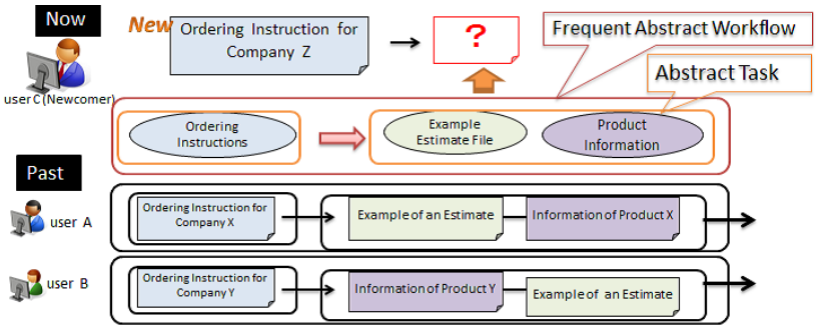


Fig. 1. Our Challenges in File Recommendation

that are similar but not identical, or that have a different ordering of elements, because the algorithm uses strict matching of elements in access sequences.

In this paper, we propose a method for recommending files and types of operations on them, that supports creative business processes as an operational tool. We consider the abstract workflows for a user. These are different from the ordinary workflows described by users, being generated by mining the file-access histories of the users. To obtain good recommendations, it is important to ignore small variations in the matching process. In this paper, we introduce for the first time the concepts of abstract files, abstract tasks, and frequent abstract workflows. The differences between collaborative filtering and our proposed method in this paper are that we add the concept of tasks to weaken the constraint of strict matching for access sequences and we apply abstraction to files, tasks, and workflows to handle newly generated files. The evaluation results indicate that our method extracts workflow patterns with longer sequences and higher support-count values (occurrence times of workflow patterns), which are more suitable as recommendations. Moreover, the results demonstrate that using the concepts of abstract tasks and abstract workflows improves the quality of the recommendations.

As a previous work, we compared several file similarity calculation methods [5]. In this paper, we focus on how to abstract tasks and workflows for recommendation.

2 Goal and Approach

Figure 1 illustrates an example of the type of workflow pattern we are investigating. We assume that user C creates a new file. The goal is to predict the files required, for recommendation to user C.

In our approach, we search for similar patterns of operation in logs and make recommendations based on them. For example, in the two patterns of user A and B, different files were accessed in different orders, but it is considered that users A and B were doing essentially the same work. Therefore, we expect to

extract an abstract pattern of work. Using this abstract workflow pattern, it becomes possible to recommend files such as "Example of an estimate" or "product information file" to user C.

While users access files located on a file server, file-access histories are stored in a log file. Each record in the log file includes timestamp, user, operation and filename information. Our method extracts abstract tasks and workflows from the log file, and reserves them in a database. The method then monitors the current file accesses by a user, and searches for workflows in the database that match the access patterns of that user to infer the user's current workflow pattern. Based on the inferred workflow, the method recommends those files, together with operations on them (e.g. open, close).

3 Definition of Workflow Model

In general, the orders of sequences of individual operations will vary, even if the outlines of the workflows are the same. To ignore such subtle differences in operational order in similar workflows, we introduce the concept of an abstract task as a group of similar combinations of file and operation, and an abstract workflow as a sequence of abstract tasks.

[Task]: A *task* is a subsequence of records in the log file. It is the basic unit in a working process's workflow. For example, from the log below, we set TGBT=3 minutes.

- Record 1: 12:00 [Open File A]
- Record 2: 12:01 [Open File B]
- Record 3: 12:10 [Open File C]
- Record 4: 13:00 [Open File D]

Three tasks will be extracted from these four records.

- Task 1: {[Open File A], [Open File B]}
- Task 2: {[Open File C]}
- Task 3: {[Open File D]}

[Abstract Task]: An *abstract task* is a set of combinations of file and operation derived by clustering similar tasks, in order to ignore small variations in between tasks. In the above example, three abstract tasks would be derived as follows.

- Abstract Task 1: [Open File-Cluster 1], [Open File-Cluster 2]
- Abstract Task 2: [Open File-Cluster 1]
- Abstract Task 3: [Open File-Cluster 3]

[Abstract Workflow]: An *abstract workflow* is a sequence of abstract tasks. For the above example, we set TGBW=30 minutes and can extract two abstract workflows that are sequences of the abstract tasks.

- Abstract Workflow 1: [Abstract Task 1] → [Abstract Task 2]
- Abstract Workflow 2: [Abstract Task 3]

[Frequent Abstract Workflow]: *Frequent abstract workflows* are groups of similar workflows, in which sequences of abstract tasks appear frequently.

4 Proposed Method

The process flow comprises two parts, namely the offline and online parts.

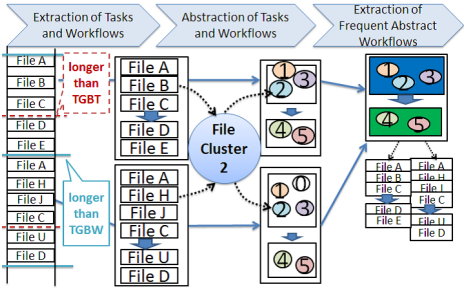


Fig. 2. Offline Part

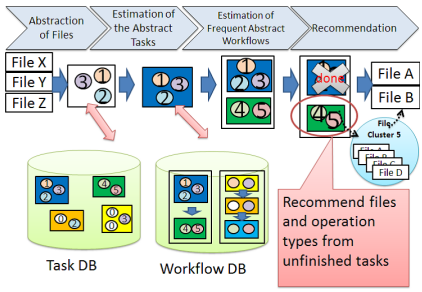


Fig. 3. Online Part

4.1 Offline Part

The aim of offline part is to extract abstract tasks and frequent abstract workflows from file-access logs.

[Extraction of Tasks and Workflow]: We first partition the log file in terms of user ID. If there are no file operations by a user during an interval longer than a certain time (TGBT for task, TGBW for workflow), we infer that the previous task/workflow had finished before the interval and the next task/workflow has started after it. We cut out tasks by parameter TGBT (150 seconds in experient) and workflows by parameter TGBW (1800 seconds in experient). In Figure 2, We partition the log into four tasks and two workflows.

[Abstraction of Tasks and Workflows]: To treat files with small differences as the same work, we want to cluster similar files used for the same purposes together. Instead of simply using the contents of the files, we calculate files' similarity based on the copy relationship and the similarity in filenames. Using the similarity between files, we apply agglomerative hierarchical clustering [2] to the files. Abstraction of files means replacing files, which appeared in the access log, with the corresponding clusters. We calculate the similarity between tasks by using the similarity between files. The similarity between tasks is a metric representing the degree of matching of two tasks in terms of file operations. It is large when two tasks have numerous abstract file operations in common. Here, we use Dice's Coefficient to calculate the degree of similarity. The formula for the similarity between $TaskA$ and $TaskB$ is

$$sim(TaskA, TaskB) = \frac{2|TaskA \cap TaskB|}{|TaskA| + |TaskB|}. \quad (1)$$

Based on the degree of similarity between tasks, we group tasks with a high degree of similarity together in a cluster as abstract tasks. We set two thresholds

to filter out small clusters and unnecessary items inside clusters. First, only when a cluster contains more than Minimum Number of Tasks (MNT, 2 in experiment) tasks, will it be treated as an abstract task. Second, only the files inside a cluster, which occurrences ratio in tasks is more than a Minimum Emergence Ratio (MER, 0.5 in experiment), will be included in the abstract task. After the task abstraction, we simply replace each task in a workflow with the corresponding abstract task to obtain the abstract workflow.

[Extraction of Frequent Abstract Workflows]: To remove any infrequent abstract workflows created, we extract those that appear frequently as frequent abstract workflows. Two parameters are used, namely the Minimum Occurrence Time (MOT, 2 in experiment) and the Minimum Sequence Length (MSL, 2 in experiment) of the workflow sequence.

4.2 Online Part

The aim of online part is to find matching abstract tasks and frequent abstract workflows in database while monitors user's current operation, and then recommends files and operations on them to the user. As mentioned in Section 3, even when users are doing the same type of work, different files are usually being handled. For this reason, we abstract files being operated on currently by the user. We then estimate the abstract task and the frequent abstract workflow (Figure 3).

[Abstraction of Files]: We abstract files by the method explained in Section 4.1. If the user is accessing a file that has a corresponding file-cluster, we simply replace the being accessed file's filename in the log by the corresponding file-cluster. However, in some cases, such as newly created files, some files do not own corresponding file-clusters. In such cases, we first replace the file being accessed by the most similar file that does have a corresponding file-cluster.

[Estimation of Abstract Tasks]: We estimate the current abstract task being operated on by the user from the abstract files being accessed. We group the abstract files being accessed into a task, and then compare this task with tasks in the database to find the most similar task.

[Estimation of Frequent Abstract Workflows]: After estimation of the abstract task, we estimate the frequent abstract workflow. Because there are several abstract workflows that contain the estimated abstract task, we score each frequent abstract workflow in the database according to these three criteria.

- Degree of matching between $workflow\,being\,accessed$ and $workflow\,in\,database$
- Frequent occurrence score for $workflow\,in\,database$
- Number of possible tasks in $workflow\,in\,database$ for recommendation

[Recommendation]: The aim of this step is to recommend files and operations based on the estimated abstract task and frequent abstract workflow. If we know about the user's current abstract task and frequent abstract workflow, we can

identify and recommend subsequent abstract tasks. However, a frequent abstract workflow is a sequence of abstract tasks, while an abstract task is a set of pairs of operations and file-clusters. Each file-cluster contains several files. The problem is how to rank the recommendation candidates. Therefore, the proposed method extracts the last access time stamp of each file, ranking files belonging to the same file-cluster by most recent access time stamp.

5 Experiments

The goal of our exteriment is to investigate the effectiveness of the features of our proposed concept, namely the abstraction of tasks and workflows. So we set up a basic method that did not use abstract tasks for comparison. The basic method simply extracts sequences of file operations as workflows and calculates those that are frequently used for recommendation.

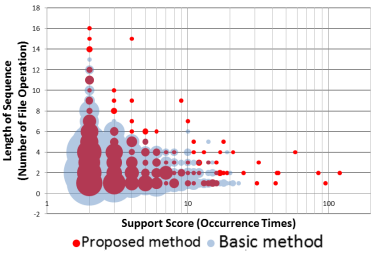


Fig. 4. Comparison of Workflow Characteristics

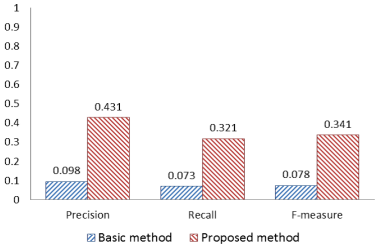


Fig. 5. Comparison of Precision, Recall, and F-measure for all Test Cases

The experimental data came from actual file-access logs provided by a commercial organization. There are 22 users in our log data. The record term is about 8 months and 9917 records and 1750 files are recorded in the log data. We split the log in a ratio of 70% to 30%. The first 70% of the log was for learning tasks and workflows and the other 30% was for evaluation. We divided the log for evaluation in terms of user IDs, using the same TGBW parameter to obtain workflows as used by the proposed method and created 151 test cases. Each test case is a workflow sequence. For each test case, the first 2 records from the beginning were input into our recommendation system to acquire a set of recommendation results. The remaining records in the test case were used as a correct answer set. By matching the examinees set and the results set, we obtained values for Precision, Recall, and F-measure. A comparison between the proposed method and the basic method was then made using the average values for all test cases.

We compared the workflows extracted by the two methods in bubble chart Figure 4. The horizontal axis represents the occurrence value (support-count value) for workflows, the vertical axis represents the number of file operations in

workflow sequences (sequence length), and the size of the bubbles represents the number of workflows. Note that basic method's bubbles are more concentrated in the lower left corner than proposed method's bubbles, which means that the workflows extracted by the basic method have smaller support-count values and shorter sequence lengths. Workflows with a small support-count value are more contingent and tend not to be reusable working patterns. In addition, workflows with small sequence lengths are undesirable because of the small amount of information available for the recommendation. Our proposed method can extract workflows with higher support-count values and greater sequence lengths than the basic method, which are more suitable for making recommendations.

We now describe the recommendation results for the evaluation experiment. Figure 5 shows the average Precision, Recall, and F-measure for all 151 test cases. Note that the proposed method performs much better than the basic method for all metrics. In particular, the F-measure score was improved from 0.078 to 0.341. The reason for this large difference is that the numbers of test cases that can be recommended are overwhelmingly different. Because the basic method can only recommend files that have been used in the past, only 25.8% of the test cases returned a result. On the other hand, because of our proposed method's abstract file operations, the proposed method can recommend files from similar file operations undertaken in the past. This enables more files to be available for recommendation. About 57.0% of the test cases returned results. We then focuses on the test cases that returned results. Our proposed method still performs considerably better than the basic method. The F-measure score was improved from 0.301 to 0.598. The reason for this is considered to be the quality of the workflows used in the recommendations. We therefore investigated the average support-count value for frequent abstract workflows that are used in the recommendations. The basic method's average support-count value for frequent abstract workflows is 4.833, while the proposed method's value is 20.186. We can conclude that the proposed method's workflows have higher support-count values and more suitable for making recommendations, which will improve the recommendation accuracy greatly.

6 Related Work

Okamoto et al. proposed a Web-page recommendation method using Web-access logs [4]. By extracting patterns in combinations of multiple attributes of the accessed pages, their method can also recommend new Web pages. Although each Web page is abstracted in their study, there is no concept of abstract tasks.

Tanaka et al. proposed a personalized document recommendation system via collaborative filtering [6]. In their system, documents viewed within a short period of time are considered to be related to the same working unit for the same purpose. They partition the access logs using the time gap between two records. We adopted this idea when extracting tasks from logs. Another file recommendation system was proposed by Lai et al. [3]. Their work proposes recommendation methods based on the knowledge-flow (KF) model. There are two differences between their work and our proposed methods. First, the aims of file abstraction

are different. In their work, files with similar topics (keywords) are grouped together. On the other side, we group files not depending on topics, but the purpose of use, which is more suitable for extracting meanful workflows. Second, their study groups similar files and then calculates KFs (workflows) directly, whereas our proposed method first groups similar files into tasks and then calculates the sequences of tasks as workflows. By introducing the concept of abstract task, our method can extract patterns with difference in order.

SUGOI [7] is for searching files using file access logs. SUGOI finds related files using file-access logs. In their study, a task is defined as the file set containing related files in simultaneous use. However, their method does not perform abstraction on tasks, which is the main difference from our study.

7 Conclusion and Future Work

In this paper, we propose a method to extract frequently used abstract-workflow patterns from the history, and to recommend files and operations by monitoring the current workflow of the target user. There are two points in our proposed method. First, our proposed method is able to extract general patterns, which are more suitable for making recommendations by abstracting such files. Second, the proposed method introduces abstract tasks to eliminate sequential relations inside tasks. The experiment results demonstrate that our proposed method is more suitable for recommendation. Consequently, the F-measure of the recommendation results was improved significantly from 0.301 to 0.598. In the future, we plan to consider a better algorithm for partitioning logs instead of simply using a fixed time. This might involve using information such as the frequency and type of operations.

References

1. Feldman, S., Duhl, J., Marobella, J.R., Crawford, A.: The hidden costs of information work. IDC WHITE PAPER (March 2005)
2. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann (2006)
3. Lai, C., Liu, D.: Integrating knowledge flow mining and collaborative filtering to support document recommendation. The Journal of Systems and Software, 2023–2037 (2009)
4. Okamoto, H., Yokota, H.: Access log based web page recommendation using multiple attributes of web pages. In: Proc. WebDB Forum 2009 (November 2009) (in japanese)
5. Song, Q., Kawabata, T., Itoh, F., Watanabe, Y., Yokota, H.: Recommendation method for files and operations based on workows of abstract tasks from access log. IPSJ (2013) (in japanese)
6. Taguchi, H., Sakojo, S., Iwata, M.: Personalized document recommendation for field engineers. DEIM (2011) (in japanese)
7. Wu, Y., Otagiri, K., Watanabe, Y., Yokota, H.: A file search method based on intertask relationships derived from access frequency and RMC operations on files. In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011, Part I. LNCS, vol. 6860, pp. 364–378. Springer, Heidelberg (2011)

Towards Addressing the Coverage Problem in Association Rule-Based Recommender Systems

R. Uday Kiran[1] and Masaru Kitsuregawa[1,2]

[1] Institute of Industrial Science, The University of Tokyo, Tokyo, Japan
[2] National Institute of Informatics, Tokyo, Japan
{uday_rage,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. Association rule mining is an actively studied topic in recommender systems. A major limitation of an association rule-based recommender system is the problem of reduced coverage. It is generally caused due to the usage of a single global minimum support (*minsup*) threshold in the mining process, which leads to the effect that no association rules involving rare items can be found. In the literature, Neighborhood-Restricted Rule-Based Recommender System (NRRS) using multiple *minsups* framework has been proposed to confront the problem. We have observed that NRRS is computationally expensive to use as it relies on an Apriori-like algorithm for generating frequent itemsets. Moreover, it has also been observed that NRRS can recommend uninteresting products to the users. This paper makes an effort to reduce the computational cost and improve the overall performance of NRRS. A pattern-growth algorithm, called MSFP-growth, has been proposed to reduce the computational cost of NRRS. We call the proposed framework as NRRS*. Experimental results show that NRRS* is efficient.

Keywords: Recommender system, association rules and coverage problem.

1 Introduction

Association rules [1] are an important class of regularities that exist in a database. Recently, these (association) rules were used in developing recommender systems [2,3]. It is because their performance is comparable to nearest-neighbor (kNN) collaborative filtering approaches, and do not suffer from scalability problems like memory-based algorithms as the rules can be mined in an offline model-learning phase [4].

A major limitation of a rule-based recommender system is the problem of **reduced coverage** [4,5]. This phenomenon is generally caused due to the usage of a single *minsup* threshold in the mining process, which leads to the effect that no rules involving rare items can be found. Fatih and Dietmar [5] have exploited the concept of rule mining with multiple *minsups* [8], and introduced NRRS (Neighborhood-Restricted rule-based Recommender System) to address the coverage problem. The NRRS uses an Apriori-like algorithm known

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 418–425, 2013.

as Improved Multiple Support Apriori (IMSApriori) [8] to generate frequent itemsets from each user's dataset. Next, it uses a tree-structure known as Extended Frequent Itemset-graph (EFI-graph) to store the rules generated from the frequent itemsets. Recommendations to the user are generated by performing depth-first search on EFI-graph. We have observed the following performance issues in NRRS:

- The NRRS is a computationally expensive recommender system. It is because of the following two reasons. First, the rules discovered with the multiple *minsups* do not satisfy the anti-monotonic property. This increases the search space, which in turn increases the computational cost of mining the frequent itemsets. Second, the IMSApriori suffers from the same performance problems as the Apriori [1], which involves generating huge number of candidate itemsets and requiring multiple database scans.
- In [10], it has been reported that EFI-graph causes NRRS to recommend uninteresting items to the users.

With this motivation, we propose an improved NRRS known as NRRS*. The key contributions of this paper are as follows:

1. We show that the frequent itemsets discovered with the multiple *minsups* framework satisfy the "convertible anti-monotonic property" [11].
2. Pei et al. [11] have theoretically shown that the search space of convertible anti-monotonic property is same as the search space of an anti-monotonic property if the frequent itemsets were discovered by a pattern-growth algorithm. Therefore, we propose a pattern-growth algorithm known as Multiple Support Frequent Pattern-growth (MSFP-growth). The MSFP-growth do not suffer from the same performance problems as the IMSApriori. As a result, it can effectively discover frequent itemsets from each user's dataset.
3. In [10], we have proposed EFI-graph++ to store the rules that satisfy the convertible anti-monotonic property. Since the rules discovered with the multiple *minsups* framework satisfy the same, we employ EFI-graph++ to store the rules and recommend products to the users [10].
4. Performance results on MovieLens dataset demonstrate that NRRS* can provide better recommendations and is more scalable than the NRRS.

The rest of the paper is organized as follows. Section 2 discusses the background and related work on association rule mining and rule-based recommender systems. Section 3 describes MSFP-growth and NRRS* algorithms. The experimental evaluations of NRRS and NRRS* have been reported in Section 4. Finally, Section 5 concludes the paper.

2 Background and Related Work

2.1 Association Rule Mining

Discovering association rules with multiple *minsups* is an important generalization proposed by Liu et al. [6] to confront the "rare item problem" that occurs

in the basic model of association rules [1]. The multiple *minsups* model of association rules is as follows [6]:

Let $I = \{i_1,\ i_2,\ \cdots,\ i_n\}$ be a set of items. Let T be a set of transactions (or a dataset), where each transaction t is a set of items such that $t \subseteq I$. A set of items $X \subseteq I$ is called an itemset (or a pattern). An itemset containing k number of items is called a k-itemset. An association rule is an implication of the form, $A \Rightarrow B$, where $A \subset I$, $B \subset I$ and $A \cap B = \emptyset$. The rule $A \Rightarrow B$ holds in T with *support* s, if $s\%$ of the transactions in T contain $A \cup B$. The rule $A \Rightarrow B$ holds in T with *confidence* c, if $c\%$ of transactions in T that support A also support B. An itemset (or a rule) is said to be **frequent** if its support is no less than the *minsup* threshold. The rules which satisfy both *minsup* and *minconf* thresholds are called **strong rules**. The *minconf* is the user-defined minimum confidence threshold value, while the *minsup* of a rule is expressed as follows: $minsup(A \Rightarrow B) = minimum(MIS(i_j) | \forall i_j \in A \cup B)$, where $MIS(i_j)$ is the user-defined *minimum item support* for an item $i_j \in A \cup B$.

An open problem with the multiple *minsups* framework is the methodology to specify the items' MIS values. Uday and Reddy [8] have tried to addressed this problem using the following methodology:

$$\begin{aligned} MIS(i_j) &= maximum(LS,\ S(i_j) - SD) \\ SD &= \lambda(1 - \beta) \end{aligned} \tag{1}$$

where, LS is the user-specified lowest minimum item support allowed, SD refers to *support difference*, λ represents the parameter like *mean*, *median* and *mode* of the item supports and $\beta \in [0, 1]$ is a user-specified constant. An Apriori-like algorithm, called IMSApriori, has been proposed to discover frequent itemsets. Researchers have also introduced algorithms based on pattern-growth technique [7,9] to discover the patterns. Unfortunately, these algorithms cannot be directly applied in NRRS to discover the frequent itemsets. The reason is that **the algorithms require user-specified items'** MIS **values**. Thus, we have proposed an alternative pattern-growth algorithm, called MSFP-growth, which can easily be incorporated in a rule-based recommender system.

2.2 Rule-Based Recommender Systems

Recently, rule-based recommender systems have received a great deal of attention in both industry and academia [2,3]. Most of these systems employ only a single *minsup* constraint to discover frequent itemsets from each user's dataset. As a result, they suffer from **coverage problem**. To confront the problem, researchers have introduced NRRS using the concept of multiple *minsup*-based frequent itemset mining [5]. The working of NRRS is as follows:

1. (**Offline phase.**) The users were grouped based on their purchased history. A transactional database is constructed using the purchased history of each user's group. Next, the complete set of frequent itemsets were discovered from each database using IMSAprori.

2. (**Online phase.**) For each user, the discovered frequent itemsets were stored in a lexical order in a tree-structure known as EFI-graph. Recommendations to the user were generated by traversing EFI-graph using depth-first search.

Since NRRS employs IMSApriori to discover frequent itemsets from each user's database, it is straight forward to argue that NRRS is a computationally expensive recommender system. Moreover, it was shown in [10] that EFI-graph can generate uninteresting recommendations to the users. In the next section, we propose NRRS*, which tries to address the performance issues of NRRS.

3 The Proposed Neighborhood-Restricted Rule-Based Recommender System*

3.1 MSFP-Growth

To reduce the computational cost of mining frequent itemsets from each user's dataset, we have investigated the types of itemsets discovered with the multiple *minsups* framework and the nature of support-difference methodology. We made the following key observations:

1. The support-difference methodology is a monotonic (or an order-preserving) function. That is, if $S(i_1) \geq S(i_2) \geq \cdots \geq S(i_n)$, then $MIS(i_1) \geq MIS(i_2) \geq \cdots \geq MIS(i_n)$.
2. For an item $i_j \in I$, the support-difference methodology specifies items' MIS values such that $S(i_j) \geq MIS(i_j) \geq LS$.
3. The frequent itemsets discovered with the multiple *minsups* framework satisfy the convertible anti-monotonic property. The correctness is based on Property 1 and shown in Lemma 1.

Property 1. (Apriori Property.) If $Y \subset X$, then $S(Y) \geq S(X)$.

Lemma 1. *(Convertible anti-monotonic property.) In a frequent itemset, all its non-empty subsets containing the item with lowest MIS must also be frequent.*

Proof. Let $X = \{i_1, i_2, \cdots, i_k\}$, where $1 \leq k \leq n$, be an ordered set of items such that $MIS(i_1) \leq MIS(i_2) \leq \cdots \leq MIS(i_k)$. Let Y be an another itemset such that $Y \subset X$ and $i_1 \in Y$. The *minsup* of X, denoted as $minsup(X) = MIS(i_1)$ $(= min(MIS(i_j)|\forall i_j \in X))$. Similarly, the *minsup* of Y, denoted as $minsup(Y) = MIS(i_1)$ $(= min(MIS(i_j)|\forall i_j \in Y))$. If $S(X) \geq MIS(i_1)$ $(= minsup(X))$, then it is a frequent itemset. Since $Y \subset X$, it turns out that $S(Y) \geq MIS(i_1))$ as $S(Y) \geq S(X) \geq MIS(i_1)$ (Property 1). In other words, Y is also a frequent itemset. Hence proved.

Based on the above observations, we introduce a pattern-growth algorithm known as MSFP-growth (see Algorithm 1). Briefly, the working of MSFP-growth is as follows:

– We construct FP-tree [11] and measure the SD value.

– Since FP-tree is constructed in support descending order of items, it turns out that the MIS value of the suffix item will be the lowest MIS value among all other items in its prefix path. Therefore, considering each item in the FP-tree as suffix item (or 1-itemset), we calculate its MIS value and consider it as "conditional minimum support" ($Cminsup$). Next, we construct its conditional pattern base, then construct its (conditional) FP-tree, and perform mining recursively on such a *tree*. The pattern-growth is achieved by the concatenation of the suffix itemset with the frequent itemsets generated from a conditional FP-tree.

Algorithm 1. MSFP-growth (T: rating transactional database, β: user-specified constant and LS: least support)

1: Scan the transactional database T and measure the support values of all items.
2: Prune the items having support values less than the LS value and sort them in descending order of their support values. Let this sorted order of items be L. Every item in F is a frequent item.
3: Let λ be the support of an representative item in the database. It generally represents mean or median of all supports of items in L. That is, $\lambda = mean(S(i_j) \forall i_j \in L)$ or $\lambda = median(S(i_j) \forall i_j \in L)$.
4: Calculate the support difference (SD) between the support and MIS of the representative item. That is, $SD = \lambda(1 - \beta)$.
5: Since support-difference methodology is an order preserving function, construct FP-tree, say $Tree$, as discussed in [11].
6: **for** each item i in the header of the $Tree$ **do**
7: calculate conditional $minsup$, $Cminsup_i = ((S(i) - SD) > LS)?(S(i) - SD) : LS)$.
8: **if** i_j is a frequent item **then**
9: generate itemset $\beta = i \cup \alpha$ with support = i.support (or $S(i)$);
10: construct β's conditional pattern base and β's conditional FP-tree $Tree_\beta$.
11: **if** $Tree_\beta \neq \emptyset$ **then**
12: call MSFPGrowth($Tree_\beta$, β, $Cminsup_i$).
13: **end if**
14: **end if**
15: **end for**

Procedure 2. MSFPGrowth(Tree, α, $Cminsup_i$)

1: **for** each i in the header of Tree **do**
2: generate itemset $\beta = i \cup \alpha$ with support = i.support.
3: construct β's *conditional pattern base* and then β's conditional FP-tree $Tree_\beta$.
4: **if** $Tree_\beta \neq \emptyset$ **then**
5: call MSFPGrowth($Tree_\beta$, β, $Cminsup_i$).
6: **end if**
7: **end for**

3.2 An Improved Neighborhood-Restricted Rule-Based Recommender System: NRRS*

The proposed NRRS* system is shown in Algorithm 3. It is a two phase algorithm involving offline and online phases. The working of NRRS* is as follows:

1. (**Offline phase.**) For each user u, top-k_1 and top-k_2, where $k_1 \geq k_2$, number of neighboring users who had similar purchased history are discovered. (The top-k_1 neighboring users will be used for learning rules, while the top-k_2 neighboring users will be used for prediction or recommendation of products to the user u.) A transactional database is generated using the purchased history of top-k_1 neighbors of the user u. Next, frequent itemsets are discovered from the database using the proposed MSFP-growth algorithm.
2. (**Online phase.**) The discovered user-specific frequent itemsets (UserFPs) of the target user and of the neighbors of the target user are used to calculate predictions using EFI-graph++ (lines 8 to 13 in Algorithm 3). The resulting confidence scores are weighted according to the similarity of the target user and the neighbor (line 14 to 16 in Algorithm 3). The similarity measure we use is Pearson correlation. These user-specific predictions are finally combined and sorted by the weighted confidence scores (line 16 and 17 in Algorithm 3). The construction of EFI-graph++ is given in [10].

Algorithm 3. NRRS* (U: set of users, $ratingDB$: rating database, k_1: learning neighborhood size, k_2: predicting neighborhood size λ: support of a representative item, β: user-specified constant and LS: least minimum item support)

1: (**Offline**: Discovery of Frequent itemsets)
2: **for** each $u \in U$ **do**
3: Let k_1-$neighborhood^u$ and k_2-$neighborhood^u$ be the set of top k_1 and k_2 neighbors for the user u in $ratingDB$. That is, k_1-$neighborhood^u = u \cup findNeighbors(u, k_1, ratingDB)$;
4: Let $ratingDB^u$ be the rating database created using the ratings given by the each user in k_1-$neighborhood^u$.
5: Let $UserFPs$ be the of frequent itemsets discovered for u in $ratingDB^u$ using MSFP-growth. That is, $UserFPs = MSFP\text{-}growth(ratingDB^u, \lambda, \beta, LS)$.
6: **end for**
7: (**Online**: Recommending items by constructing EFI-graph for each user)
8: **for** each user $u \in U$ **do**
9: $recommendedItems = \emptyset$;
10: **for** each user $u_i \in k_1$-$neighborhood^u$ **do**
11: $userRecs = Recommend(u, EFIG++(UserFPs(u_i)))$;
12: Scan $ratingDB^u$ to measure the support count of infrequent itemsets in EFI-graph of u;
13: weightedUserRecs = adjustConfidenceScoresBySimilarity($userRecs, u, u_i$);
14: $recommendedItems = recommendedItems \cup weightedUserRecs$;
15: **end for**
16: $recommendedItems = sortItemsByAdjustedScores(recommendedItems)$;
17: output $recommendedItems$;
18: **end for**

4 Experimental Results

In this section, we evaluate the NRRS and proposed NRRS*. The programs are written in java and run with Ubuntu on a 2.66 GHz machine with 2GB

memory. The experiments were pursued on "MovieLens" rating dataset consisting of 100,000 ratings provided by 943 users on 1,682 items.

In order to test NRRS and NRRS* frameworks also in settings with low data density, we varied the density level of the original data sets by using sub-samples of different sizes of the original data set. Four-fold cross-validation was performed for each data set; in each round, the data set was split into a 80% training set and a 20% test set.

To compare the predictive accuracy of NRRS and NRRS* systems, we use the following procedure. First, we determine the set of existing "like" statements (ELS) in the 20% test set and retrieve a top-N recommendation list with each system based on the data in the training set. The top-N recommendations are generated based on the confidence of the producing rule. The set of predicted like statements returned by a recommender shall be denoted as Predicted Like Statements (PLS). The standard information retrieval accuracy metrics are used in the evaluation. *Precision* is defined as $\frac{|PLS \cap ELS|}{|PLS|}$ and measures the number of correct predictions in PLS. *Recall* is measured as $\frac{|PLS \cap ELS|}{|ELS|}$, and describes how many of the existing "like" statements were found by the recommender. The averaged precision and recall values are then combined in the usual *F-score*. That is, $F = 2 \times \left(\frac{precision \times recall}{precision + recall}\right)$.

Table 1 shows the performance of NRRS and NRRS* systems on MovieLens dataset with varying densities. It can be observed that the performance of NRRS* is relatively better than NRRS, independent of the density. The reason is that EFI-graph++ facilitated NRRS* to recommended only those items that had high confidence value although the past transactions of a user represented an infrequent itemset.

Table 1. Performance comparison of NRRS and NRRS* at different density levels

Density		10%	30%	50%	70%	90%
Movie Lens	F1	35.75	47.75	57.89	60.91	62.72
	Precision	45.9%	60.23%	63.61%	64.7%	63.8%
	Recall	29.3%	39.56%	53.12%	57.54%	61.69%

Fig. 1 shows the runtime taken by NRRS and NRRS* systems to generate recommendations to the users with varying dataset sizes for learning the rules

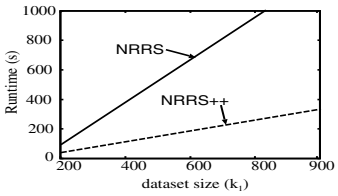


Fig. 1. Runtime comparison of mining frequent itemsets for each user in NRRS and NRRS*. The k_1 represents a users' dataset size (or neighborhood size).

(i.e., k_1). It can be observed that although both algorithms are linearly scalable with increase in dataset size, NRRS* is relatively more efficient than NRRS system. It is because the MSFP-growth discovered frequent itemsets for each users' dataset more effectively than IMSApriori. Another important observation is that NRRS* is more scalable than NRRS with increase in k_1 neighborhood (or dataset) size for learning the rules.

5 Conclusions

This paper has proposed an effective and efficient rule-based recommender system, called NRRS*, to confront the coverage problem that persists in the conventional rule-based recommender systems. It has also shown that frequent itemsets discovered with the multiple *minsups* framework satisfy the convertible anti-monotonic property, and therefore, can be effectively discovered using the proposed pattern-growth algorithm, called MSFP-growth. Experimental results demonstrate that NRRS* is efficient than NRRS.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207–216 (1993)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749 (2005)
3. Smyth, B., McCarthy, K., Relly, J., O'Sullivan, D., McGinty, L., Wilson, D.C.: Case Studies in Association Rule Mining for Recommender System. In: IC-AI, pp. 809–815 (2005)
4. Lin, W., Alvarez, S.A., Ruiz, C.: Efficient Adaptive-Support Association Rule Mining for Recommender Systems. Data Mining and Knowledge Discovery 6(1), 83–105 (2002)
5. Gedikli, F., Jannach, D.: Neighborhood-restricted mining and weighted application of association rules for recommenders. In: Chen, L., Triantafillou, P., Suel, T. (eds.) WISE 2010. LNCS, vol. 6488, pp. 157–165. Springer, Heidelberg (2010)
6. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: KDD, pp. 337–341 (1999)
7. Hu, Y.-H., Chen, Y.-L.: Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. Decision Support Systems 42(1), 1–24 (2006)
8. Uday Kiran, R., Krishna Reddy, P.: An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules. In: Proceedings of CIDM, pp. 340–347 (2009)
9. Uday Kiran, R., Krishna Reddy, P.: Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In: Proceedings of EDBT, pp. 11–20 (2011)
10. Uday Kiran, R., Krishna Reddy, P.: An Improved Neighborhood-Restricted Association Rule-based Recommender System Using Relative Support. To be Appeared in ADC (2013)
11. Pei, J., Han, J.: Constrained frequent pattern mining: a pattern-growth view. SIGKDD Explor. Newsl. 4(1), 31–39 (2002)

Opinion-Based Collaborative Filtering to Solve Popularity Bias in Recommender Systems

Xiangyu Zhao[1], Zhendong Niu[1], and Wei Chen[2,1]

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2] Agricultural Information Institute, Chinese Academy of Agricultural Sciences, Beijing, China

Abstract. Existing recommender systems suffer from a popularity bias problem. Popular items are always recommended to users regardless whether they are related to users' preferences. In this paper, we propose an opinion-based collaborative filtering by introducing weighting functions to adjust the influence of popular items. Based on conventional user-based collaborative filtering, the weighting functions are used in measuring users' similarities so that the effect of popular items is decreased with similar opinions and increased with dissimilar ones. Experiments verify the effectiveness of our proposed approach.

Keywords: recommender system, popularity bias, collaborative filtering.

1 Introduction

Recommender systems have gained considerable development to help people in retrieving potentially useful information in the current age of information overload [2][4][7]. However, there are still many disadvantages. The "Harry Potter Problem" [1] is one of them. Harry Potter is a runaway bestseller, which always appears in customers' recommendation list whatever books they are browsing. That is, recommended items are biased towards popular, well-known items. Thus, the recommendations may not be related to users' preferences. Furthermore, users are already likely to have made conscious decisions regarding these popular items. Therefore, this kind of recommendations rarely lead users to additional purchases as they are lack of novelty [11]. Some studies have been done on solving this popularity bias problem [1][3][9][11]. However, there is no systematic discussion of popularity bias in these studies.

In this paper, by analyzing the cause and influence of popularity bias and comparing some solutions, an assumption is proposed that a user's rating on an item means differently to his/her preference according to the popularity of the target item. Based on the assumption, approaches are proposed to improve recommendation by weighting items based on their effects to the user model according to their

[1] http://glinden.blogspot.com/2006/03/early-amazon-similarities.html

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 426–433, 2013.

popularities. Based on conventional user-based collaborative filtering (UserCF), the weighting functions are used for measuring users' similarities. The main contributions of the proposed work are as follows: we introduce popularity bias formally and analyze the cause and influence; two weighting functions and an approach is proposed to solve popularity bias; experiment results show that our approach outperforms conventional ones on both accuracy and diversity.

2 Related Work

Recommendation techniques have been studied for several years. Collaborative filtering (CF) is a popular one, since it is not necessary to analyze the content of the candidate items and uses collective intelligence instead. UserCF is a typical CF approach, which is based on a basic assumption that people who agreed in the past tend to agree again in the future. The level of agreement can be measured by similarity between users. There are several functions to measure the similarity. In this paper, Cosine similarity is chosen to measure user similarity. Based on the similarity calculation, a set $N(u)$ of the nearest neighbors of user u is obtained. After that, ratings for unknown items are predicted by adjusted weighted sum of known ones, and items with high predicted values are recommended [5]. The predicted rating of item i by user u can be calculated as:

$$R^*(u,i) = \overline{R(u)} + \frac{\sum_{a \in N(u)} sim(u,a) \cdot (R(a,i) - \overline{R(a)})}{\sum_{a \in N(u)} |sim(u,a)|} \tag{1}$$

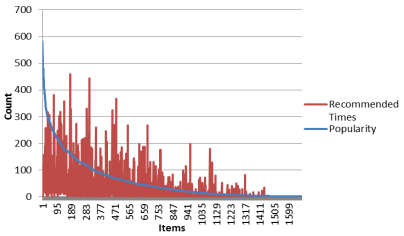
where $R(u,i)$ is the rating value that user u rates item i, and $\overline{R(u)}$ represents the average rating of user u.

Several researches have been worked out to solve popularity bias in recommender systems. Adomavicius et al. use a popular recommendation technique to predict unknown ratings, and then consider only the items that are predicted above the pre-defined rating threshold as candidate recommendations. Among these candidate items, they recommend popular ones to gain high precision, or unpopular ones to gain high diversity. The rating threshold is used to trade off precision against diversity by controlling a re-ranking level [1]. Lai et al. add a punishing function to popular items in item-base CF (ItemCF) to reduce the chance of popular items to be recommended in order to improve Error Rate (ItemCF-PP) [9]. Breese et al. use inverse user frequency (IUF) to reduce the effect of popular items in UserCF (UserCF-IUF) [3]. However, experiment results show that the IUF weighting does not improve the performance from the baseline method that does not use any weighting for items [8]. In the opinion of Oh et al., people have different preferences in terms of the popularity of items, which is defined as Personal Popularity Tendency (PPT) [11]. They recommend items similar to the PPT of each individual while reflecting active users' preferences as much as possible. Many of the above solutions simply change the popularity distribution of the recommendations [1][9][11]. We tend to solve popularity bias by utilizing the cause and influence of popularity bias, which will be introduced detailed next.

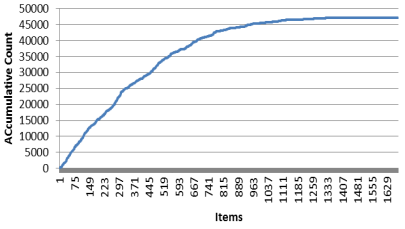
3 Popularity Bias in Recommender Systems

3.1 The Phenomenon of Popularity Bias

The popularity bias problem can be described by Matthew Effect. It is a phenomenon where "the rich get richer and the poor get poorer" [10]. This phenomenon can be seen in Fig. 1[2], intuitively. The blue line in Fig. 1(a) represents the item rating frequencies in the MovieLens dataset. The red one represents the recommended times by UserCF for the target items in a top-50 recommendation task. We count the accumulative recommended times for items which are more popular than the target one. Figure 1(b) shows the result, which indicates the rising rate of the accumulative recommended times. It is obvious that popular items are recommended much more frequently than unpopular ones, which verifies the phenomenon of popularity bias in a certain degree.



(a) Items' rating frequencies and recommended times.



(b) Accumulative count of recommended times for items which are more popular than the target one.

Fig. 1. Items' rating frequencies and recommended times in the MovieLens dataset

Since the increase in popularity of the items is affected by Matthew Effect, the indication of users' preferences from the feedbacks on popular items is magnified. On the contrary, the unpopular items in the long tail get little attention and feedbacks, and are hardly recommended. Every feedback of them is based on users' individual search. Therefore, the difficulty of getting a target item is relevant to the popularity of it. Because of the difficulty of finding, the feedbacks for unpopular items can be recognized as great interest. We formalize this idea as the following hypothesis.

Hypothesis 1. *In an information system that satisfies the following requirements:*

The system provides recommendations[3] *to help people in solving the problem of information overload; a user can browse an item in the system coming from*

[2] Items are in descending order according to the number of ratings.

[3] The recommendations mentioned here include items which are recommended by advertisements, word-of-mouth, and recommender systems.

either the recommendation of the system or his/her individual search, and can give it a feedback after browsing.

The following statements are true:

A feedback is an indication of the target user's preference; the level of the preference is relevant to the difficulty of finding the target item.

A simple way to solve the popularity bias problem is to reduce the chance of popular items being recommended. However, the chance of users' preferred popular items being recommended is decreased simultaneously in this way. According to Hypothesis 1, popularity bias leads that items with different popularity indicate different levels of users' preference. As a result, to distinguish the effect of the items with different popularity while modeling users' preferences may be a good solution to decrease the effect of Matthew Effect on popularity bias.

3.2 Methods to Deal with Popularity Bias

According to Hypothesis 1, we believe that items with different popularity indicate different levels of users' preferences, which is similar to Breese's opinion [3]. Different from conventional UserCF, which consider that the item effect is the same in a user model, we try to measure the effect of items based on the popularity to solve the popularity bias problem. The baseline method is to distinguish item effect with a weighting function according to item popularity. IUF [3] is an example of the weighting function. It can be found that there are two characteristics in the IUF weighting function: the weights of the most popular items are only a little less than the weights of the least popular ones (no more than 20%); the weights decrease much faster at the head than at the long tail.

According to Hypothesis 1, the level of the preference to an item is relevant to the difficulty of finding it. The recommended times for items can indicate the difficulty of finding them to a certain degree. Figure 1(b) shows that the recommended times for items meet the following conditions: the recommended times for items are inversely proportional to their popularity; the recommended times for items are almost the same at the head[4]; the recommended times for items between head and long tail are different considerably.

Obviously, the characteristics of IUF do not match these conditions. Therefore, IUF may not be suitable to solve popularity bias problem, which is verified in Breese's experiments [3]. To match the difficulty of finding items, a *Base* weighting function is proposed, which can be written as:

$$w_{\text{Base}}(i) = \frac{1}{\log(p_i)} \tag{2}$$

where p_i represents the popularity of item i. It can be found that the *Base* function meet the above conditions of recommended times for items. Other functions that meet the conditions may be suitable for weighting the effect of items. We leave exploring which is the best one to future work.

[4] The slope in Fig. 1(b) is almost the same at the head, which indicates that the recommended times are almost the same for those popular items.

However, the common situation is that only the items at the head are affected by the Matthew Effect, thus get more and more popular. Therefore, fair treatment for all items is not appropriate. The items are divided into two subsets by a threshold T_P. Only the head set of items are adjusted by a weighting function. This is the *Popular* function, which can be written as:

$$w_{\text{Popular}}(i) = \begin{cases} \frac{1}{\log(p_i)} & , p_i > p_{\text{H}} \\ 1 & , p_i \leq p_{\text{H}} \end{cases} \tag{3}$$

where p_{H} represents the minimal popularity in the head set.

3.3 Improving Recommendation Approaches

Our improving algorithm is based on conventional UserCF. In order to deal with the popularity bias problem of UserCF, an intuitive solution is to reduce the influence of the popular items when building user models. As a result, the similarities between users in improved UserCF can be measured as:

$$sim_{\text{I}}(u,a) = \frac{\sum_{i \in I(u,a)} w(i) \cdot R(u,i) \cdot R(a,i)}{\sqrt{\sum_{i \in I(u)} \left(w(i) \cdot R(u,i)\right)^2} \cdot \sqrt{\sum_{i \in I(a)} \left(w(i) \cdot R(a,i)\right)^2}} \tag{4}$$

where $I(u)$ is the item set that rated by user u, $I(u,a)$ is the item set that co-rated by both user u and user a. The weighting function $w(i)$ can be either the *Base* function or the *Popular* one. The similarity function is used in Weighting-UserCF (WUserCF) when calculating predicted ratings according to (1).

However, simply reducing the influence of all the popular items may not be suitable for measuring users' similarities. A popular item co-rated by both two users does not mean the same as a popular item only rated by one user. The latter one is much more significant than the former. As a result, the effect of the latter one should be enhanced. The Co-rated-based Weighting-UserCF (CWUserCF) is proposed based on this idea. The weighting function of it can be written as:

$$w_{\text{C}}(i) = \begin{cases} w(i) & , i \in I(u,a) \\ \frac{1}{w(i)} & , i \notin I(u,a) \end{cases} \tag{5}$$

This function replaces the original weighting function in measuring the similarity function for CWUserCF according to (4). The CWUserCF uses this similarity function to predict ratings according to (1).

However, both WUserCF and CWUserCF do not consider the rating values which indicate the opinion of the target user. A similar idea to CWUserCF to deal with the problem is that the influence of popular items with different opinions should be increased while the ones with the same opinion should be decreased. The weighting function of the Opinion-based Weighting-UserCF (OWUserCF) can be written as:

$$w_{\text{O}}(i) = \begin{cases} w(i) & , i \in S(u,a) \\ \frac{1}{w(i)} & , i \notin S(u,a) \end{cases} \tag{6}$$

where $S(u,a)$ is the item set that user u and user a have similar opinions on. The items in $S(u,a)$ meet the condition which is described as:

$$(R(u,i) - \overline{R(u)}) \cdot (R(a,i) - \overline{R(a)}) > 0 \tag{7}$$

The weighting function considers the opinion on the target item. Correspondingly, the similarity function should consider the opinion, too. As a result, the similarity function can be written as:

$$sim_{\mathrm{O}}(u,a) = \frac{\sum_{i \in I(u,a)} w_{\mathrm{O}}(i) \cdot (R(u,i) - \overline{R(u)}) \cdot (R(a,i) - \overline{R(a)})}{\sqrt{\sum_{i \in I(u)} \left(w_{\mathrm{O}}(i) \cdot (R(u,i) - \overline{R(u)})\right)^2} \cdot \sqrt{\sum_{i \in I(a)} \left(w_{\mathrm{O}}(i) \cdot (R(a,i) - \overline{R(a)})\right)^2}} \tag{8}$$

OWUserCF uses this similarity function to predict ratings according to (1).

4 Experiment

4.1 Experiment Setup

Experiments are carried out on the MovieLens dataset, which consists of 100,000 ratings which are assigned by 943 users on 1682 movies. Collected ratings are in a 1-to-5 star scale. We use 5-fold cross validation for the evaluation. For each data split, 80% of the original set is included in the training set to generate recommendations, and 20% of it is included in the test one to evaluate the recommendation results.

The proposed approaches are evaluated in the experiment, comparing with two baseline approaches (UserCF and ItemCF) and two existing popularity bias solutions (ItemCF-PP [9] and UserCF-IUF [3]). All these approaches are evaluated with one accuracy metric, the Normalized Discounted Cumulative Gain (NDCG) [6], and two diversity metrics, coverage (COV) [1] and coverage in long tail (CIL),comparing with NDCG+[5].

The size of nearest neighbors for UserCF, UserCF-IUF, WUserCF, CWUserCF, and OWUserCF is 50 (an empirical value); the value of T_P is 0.1, which is the best one while analyzing the impact of the threshold.

4.2 Experiment Results

Table 1 illustrates the results of the proposed approaches and the comparative ones in top-N recommendation task on MovieLens dataset. For each column in Table 1, we highlight the top 3 best performed methods.

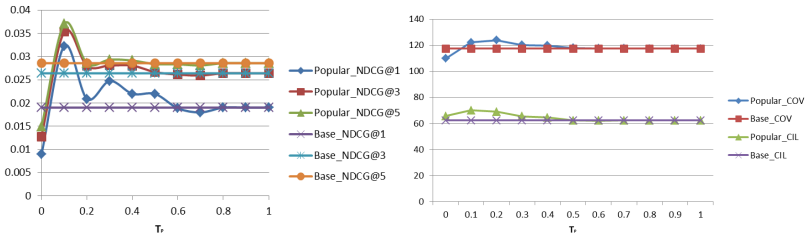
As can be seen from the results, ItemCF and ItemCF-PP get the best diversity and the worst accuracy, and the accuracy results are much less than others. Without considering these two approaches, OWUserCF performs best on both accuracy and diversity. However, WUserCF and CWUserCF do not effectively improve the baseline approaches especially on accuracy metrics. It shows that

[5] NDCG+ is the NDCG result while ranking the items in test set.

they do wrong when decreasing the effect of popular items with different opinions which means significant in measuring the user similarities.

The results indicate that popularity bias exists in UserCF approach, as OWUserCF can improve the recommendation quality a lot. However, the effect of popular items cannot be decreased simply since WUserCF and CWUserCF do not get effective improvement. It is necessary to distinguish whether the users have similar opinions on the target popular item. If the answer is yes, it indicates that these two persons have similar opinions as most people do, and then the effect can be decreased. Otherwise, the effect can be increased as its significant dissimilar idea. That is the reason why our proposed OWUserCF method, which takes opinion direction into account, achieves the best results in the experiment.

In OWUserCF, *Popular* is a better weighting function than *Base*. This verifies our idea that just the effect of very popular items should be adjusted. A new experiment is conducted to analyze the impact of the value of T_P. Figure 2 shows the change in performances when T_P increases from 0 to 1, step by 0.1. It can be easily found that when T_P=0.1, OWUserCF gains the best performances.



(a) NDCG (b) COV and CIL

Fig. 2. Performance as a function of the value of threshold T_P

Table 1. Performance of different approaches

	NDCG			COV@5	CIL@5	NDCG+		
	1	3	5			1	3	5
UserCF	**0.0089**	**0.0128**	**0.0148**	110	66	**0.74**	**0.73**	**0.73**
ItemCF	0.0005	0.0006	0.0007	**156**	**142**	0.56	0.61	0.65
UserCF-IUF	0.0068	0.0097	0.0132	95	58	**0.74**	**0.73**	**0.73**
ItemCF-PP	0.0005	0.0003	0.0006	**142**	**139**	0.38	0.39	0.40
WUserCF-Base	0.0048	0.0087	0.0128	97	57	0.73	0.72	0.72
WuserCF-Popular	0.0023	0.0096	0.0103	110	67	0.70	0.69	0.70
CWUserCF-Base	0.0063	0.0087	0.0126	88	52	0.73	**0.73**	0.72
CWuserCF-Popular	0.0040	0.0064	0.0082	98	61	0.73	0.71	0.71
OWUserCF-Base	**0.0190**	**0.0264**	**0.0286**	115	62	**0.77**	**0.77**	**0.77**
OWuserCF-Popular	**0.0323**	**0.0353**	**0.0370**	**122**	**70**	0.72	**0.74**	**0.75**

5 Conclusions and Future Work

In this paper, we propose an opinion-based collaborative filtering approach to solve popularity bias in recommender systems. Experiment results show that our proposed approach (OWUserCF) outperforms the baseline and comparative ones on both accuracy and diversity. This verifies the effectiveness of our proposed approach as well as the existence of popularity bias in recommender systems.

This work analyzes and verifies the popularity bias of recommender systems. In our subsequent research, we will explore other functions to weight the effect of items, and judge which one is more suitable. Furthermore, we will do more experiments using other datasets to further evaluate our proposed approach.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Project Nos. 61250010, 61003263, and 61272361).

References

1. Adomavicius, G., Kwon, Y.: Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. IEEE Trans. on Knowl. and Data Eng. 24(5), 896–911 (2012)
2. Bahmani, A., Sedigh, S., Hurson, A.: Ontology-based recommendation algorithms for personalized education. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012, Part II. LNCS, vol. 7447, pp. 111–120. Springer, Heidelberg (2012)
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Cooper, G.F., Moral, S. (eds.) Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI 1998), pp. 43–52. Morgan Kaufmann Publishers Inc., San Francisco (1998)
4. Chen, W., Niu, Z., Zhao, X., Li, Y.: A hybrid recommendation algorithm adapted in e-learning environments. Journal of World Wide Web, 1–14 (2012)
5. Delgado, J., Ishii, N.: Memory-based weighted-majority prediction for recommender systems. In: ACM SIGIR 1999 Workshop on Recommender Systems: Algorithms and Evaluation (1999)
6. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. 20(4), 422–446 (2002)
7. Jia, D., Zeng, C., Nie, W., Li, Z., Peng, Z.: A new approach for date sharing and recommendation in social web. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012, Part II. LNCS, vol. 7447, pp. 314–328. Springer, Heidelberg (2012)
8. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and development in Information Retrieval (SIGIR 2004), pp. 337–344. ACM, New York (2004)
9. Lai, S., Xiang, L., Diao, R., Liu, Y., Gu, H., et al.: Hybrid Recommendation Models for Binary User Preference Prediction Problem. In: KDD-CUP 2011 Workshop (2011)
10. Merton, R.K.: The Matthew Effect in Science. Science 159(3810), 56–63 (1968)
11. Oh, J., Park, S., Yu, H., Song, M., Park, S.: Novel Recommendation based on Personal Popularity Tendency. In: Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2011), pp. 507–516 (2011)

Exploring Trust to Rank Reputation in Microblogging

Leila Weitzel[1,2], José Palazzo Moreira de Oliveira[2], and Paulo Quaresma[3]

[1] Federal University of Pará, Pará, 68501-970, BR
lmartins@ufpa.br
[2] Federal University of Rio Grande do Sul, Rio Grande do Sul, 91501-970, BR
palazzo@inf.ufrgs.br
[3] University of Évora, Évora, 7000, PT
pq@uevora.pt

Abstract. The Web 2.0 is the top manifestation of User-Generated Content systems, such as reviews, tags, comments, tweets etc. Due to their free nature such systems contain information of different quality levels. Consequently, it is difficult for users to determine the quality of the information and the reputation of its providers. The quality evaluation is a matter of great concern, especially in medical and healthcare domain. To help the posts quality assessment this paper presents an alternative to measuring reputation based on social interactions. As a test case, we explore the data structure of the Twitter micro blogging service. Our main contribution is to provide a new methodology to Rank Reputation in a network structure based on weighted social interaction. This approach can guide Internet users to encounter authority and trustworthy sources of online health and medical information in Twittershpere. The results show that the rank methodology and the network structure have succeeded in inferring user reputation.

Keywords: Social Network Analysis, Twitter, Reputation, Quality information.

1 Introduction

The number of individuals using the Internet to acquire health care information is constantly increasing. The most common reasons that stimulate this behavior are connected with the need to try to find information or advice on health condition, symptoms, diseases, or treatment [1]. Looking on the bright side, these searches are intended to elicit discussion and communication between patient and the primary care physician. Looking at the negative side, incorrect information could be life-threatening [2]. Anyone can post anything on the web, regardless of his or her background or medical qualifications. The Web 2.0 is the strongest manifestation of User-Generated Content systems – UGC – and as such supports more potential for growth than any other form of content on the Web today. Popular UGC systems domains include blogs and web forums, social bookmarking sites, photo and video sharing communities, as well as social networking platforms such as Twitter, Facebook and MySpace. The main challenge posed by content in social media sites is the fact

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 434–441, 2013.

that the distribution of quality has high variance: from very high-quality items to low-quality. Due to their nature, such systems contain information of different quality levels and makes it difficult for users to determine the quality of the information and the reputation of its providers [3].

Based on the context described, this paper proposes measuring reputation based on social interactions. As a test case we explore the rich data structure of Twitter microblogging service. Our main contribution in this research is a new methodology to Rank Reputation in a new network structure based on weighted social interaction. This approach can guide Internet users to meet authority and trustworthy sources of online health and medical information in Twittershpere. In our study, user's reputation implies that health information disseminated within their social network is thought to be credible and worthy of belief. We focus on Twitter since it provides a large amount, a diversity and varying quality of content. To meet the objective of the proposal, we modeled the social interactions in a graph-based retweet weighted ties - a retweet is a forward of a tweet. To the best of our knowledge, this is the first study that uses the retweet mechanism as interaction tool to infer Reputation. The obtained results shown that the new rank methodology and the developed network structure have succeeded in inferring user reputation.

2 Social Network: Microblogging Main Features

Social Network Analysis – SNA – is the mapping and measuring of relationships and flows between people, groups, organizations, and other connected information or knowledge entities. A social network is a social structure between actors, individuals or organizations [4]. People now connect with other people beyond geographical and time barriers, diminishing the constraints of physical and temporal boundaries in creating new ties. These ties can characterize any type of relationship, friendship, authorship, etc. The Microblogging services have rapidly developed as a recently emerging service because of its timeliness, convenience. This singular social environment has received a considerable attention from academic researchers, because it represents a fresh medium for search and diffusion of information [5]. Twitter users post tweets and if other users like or find its content truly interesting they repost it or "retweet it". "Retweeting" is a key mechanism for information diffusion in microblogging. By allowing "Twitterers" to forward information that they estimate interesting, important, or entertaining the retweeting process behaves just like an informal recommendation system. It is believed that, when someone "retweet" your post, they gives you a kind of reputation by sharing your post with their own followers or contacts.

3 Proposed Model and Method

Latest studies show that not only the network structural characteristics identify the user's importance and also the user's communication activity as the exchange of information via messages, posts, comments [6-9]. We consider that the retweet function

is likely to be interpreted as a form of endorsement for both the message and the originating user. Retweet function represents the degree or level of interactions between users, forging trust-based relations. In this manner, we created a network structure based on retweet weighted ties named Retweet-Network or simply ***RT-network***. Up to our knowledge, this is the first time that this network structure is modeled. We model the ***RT-network*** as a direct weighted graph $\mathbf{G_{RT}} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ with the following properties:

1. The set of nodes (denoting the set of users) $\mathcal{V} = \{v_1, v_{2,\ldots}\}$
2. The set of edges (representing retweet function) $\mathcal{E} = \{e_1, e_2, \ldots\}$ and
   If $\exists$ edge $e_k = (v_i, v_j) \in \mathcal{E}$, from v_i to v_j this means that user v_i "retweet" user v_j. The user v_i is denoted as source user, and v_j as target user.
3. The set of weights (characterizing the strength of trust ties) $\mathcal{W} = \{w_1, w_2, \ldots\}$
   The $w(e_k)$ is a function defined for edges as follows:

$$w(e_k) = \left(\frac{\sum \mathbf{RT}_{v_j}}{\mathbf{RT_{total}}}\right) + \alpha \tag{1}$$

Where the parameter $\sum \mathbf{RT}_{v_j}$ is the counted retweets for v_j from a specific v_i, and $\mathbf{RT_{total}}$ is the total number of retweet of a target user. This fraction denotes how much a user trusts a particular target user. The parameter α is a sort of discount rate representing relationships (follower, following and friendship) between source users and target users. Acknowledged celebrities attract thousands of followers such as Lady Gaga, Britney Spears, and Ashton Kutcher etc. For example, Ashton Kutcher is a classic Twitter celebrity phenomenon; he has almost five million followers. By default, the tweets and retweet are broadcasted within their network; consequently this behavior overestimates the reputation measure. The parameter α is estimated as: α = 0.1 If a user v_i is a follower of v_j, since the retweet function is the occurrence that is expected to happen, thus α has lower weight, and α = 0.9 in all other cases, thus α has higher weight.

3.1 Ranking Reputation Approach – ($\mathcal{R}$a$\mathcal{R}$)

Perhaps the most frequently used centrality measures are Degree - $\mathcal{D}c$, Closeness - $\mathcal{C}c$, Betweenness - $\mathcal{B}c$, and Eigenvector - $\mathcal{E}c$. $\mathcal{B}c$ is based on the shortest paths between nodes and focuses on the number of visits through the shortest path. In a directed graph, for a vertex v, we denote the In-Degree $d_{in}(v)$ as the number of arcs to v and the Out-degree $d_{out}(v)$ as the number of arcs from it, thus $\mathcal{D}c$ is $d_{in} + d_{out}$. The $\mathcal{C}c$ measures how close a vertex is to all other vertices in the graph. Nodes with high values on $\mathcal{E}c$ are linked to well-connected nodes and so may influence many others in the network either directly or indirectly through their connections [10]. PageRank is another common measure, it is generally used to rank WebPages and ultimately to rank "popularity". It is defined formally as the stationary distribution of a stochastic process whose states are the nodes of the web graph, it computes the rank of websites by the number and quality of incoming links [11]. In

order to address the goal of this work and based on the summary descriptions above, we describe the **Ra**nk **R**eputation approach - $\mathcal{RaR}_{v_j}$ combining weighted centralities measures that best fit node reputation as follows:

$$\mathcal{RaR}_{v_j} = \frac{\sum_{i=1}^{n}(\mathbb{m}_i * \mathbb{w}_i)}{\mathbf{Max}\{(\mathbb{m}_i * \mathbb{w}_i),..,(\mathbb{m}_n * \mathbb{w}_n)\}} \text{ where } 0 < \mathcal{RaR}_{v_j} < 1, \quad \sum_{i=1}^{n} \mathbb{w}_i = \mathbf{1}, \quad j = \mathbf{1},..,\mathbf{m}$$

We model $\mathcal{RaR}_{v_j}$ as a function of $(\mathbb{M}, \mathbb{W})$ with the following properties: $\mathbb{M} = \{\mathbb{m}_1, .., \mathbb{m}_n\}$ is a set of centrality measures, such as: $\{\mathcal{D}c, \mathcal{B}c, \mathcal{C}c, \mathcal{E}c, \text{Prank}, d_{\text{in}}, d_{\text{out}}\}$ and $\mathbb{W} = \{\mathbb{w}_1, \ldots, \mathbb{w}_n\}$ is a set of non-negative weights.

Therefore, given the input directed weighted graph $\mathbf{G}_{RT} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, the $\mathcal{RaR}_{v_j}$ is computed iteratively. In the first step, for each node v_j it is calculated the metrics $\mathbb{m}_i \in \mathbb{M}$. In second step, it is set out arbitrarily the weight $\mathbb{w}_i \in \mathbb{W}$, the estimated weights must follow the condition $\forall \mathbb{w}_i \in \mathbb{W}$, $\sum_{i=1}^{n} \mathbb{w}_i = \mathbf{1}$, where $\mathbb{w}_i \subset \{\mathbf{0.0}, \mathbf{0.1}, \mathbf{0.2}, \ldots \mathbf{1.0}\}$, hence, it is possible that $\exists\, \mathbb{w}_k = \mathbf{0} \mid \mathbb{m}_k * \mathbb{w}_k = \mathbf{0}$. In the third step, for each node v_j it is computed $\sum_{i=1}^{n}(\mathbb{m}_i * \mathbb{w}_i)$, thereafter compute $\mathcal{RaR}_{v_j}$, assigning a label to this measure as $\mathcal{RaR}^{\varphi}_{v_j}$. In the fifth step, we calculate the **AvP** ($\mathcal{RaR}^{\varphi}_{v_j}$) - the Average Precision measure (as described below). In the last step, return de best fit; return $\mathcal{RaR}^{\varphi}_{v_j}$ that achieved the highest **AvP**. In an optimal ranked retrieval system, a set of relevant retrieved documents are given by the top k retrieved documents. Thus, AvP measure is often used as an indicator for evaluating ranked retrieval results [12]. We considered relevant documents, i.e., relevant users, those that are public healthcare. We select this alternative mostly because, if the site receives funding from commercial firms or private foundations, then the financial dependence has the potential to bias the information presented. For instance, if the purpose of the information is primarily to sell a product, there may be a conflict of interest since the manufacturer may not want to present findings that would discourage you from purchasing the product. In order to gain insight about our rank approach, we utilize the conclusions of Kwak and Cha [5, 13]. Kwak and colleagues [5] proposed popularity measures that are based on followers count (M1) and PageRank (M2). Cha et al [13] use Indegree (M3) to infer user influence. We also modeled a binary network, named RT-Binary to evaluate the quality of proposed approach. The RT-Binary network was modeled also as a directed graph $\mathbf{G}_{\mathbf{RT}} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, with the following properties:

1. The set of nodes (denoting the set of users) $\mathcal{V} = \{v_1, v_2, \ldots\}$
2. The set of edges (representing retweet function) $\mathcal{E} = \{e_1, e_2, \ldots\}$ and
   If $\exists$ edge $e_k = (v_i, v_j) \in \mathcal{E}$, **i.e.,** from v_i to v_j this means that user v_i "retweet" user v_j
3. The set of weights $\mathcal{W} = \{w_1, w_2, \ldots\}$ and $w(e_k) = 1$

We analyzed 152 user profiles randomly chosen and their respective retweets. The data was acquired from those who reported to have some interest in health subject,

during the period of March/April 2011. From these seed users' accounts, we reached 4350 retweets and 1232 user account to build the two data sets.

4 Main Results

The Table 1 illustrates the best-fit results according to the rank reputation algorithm. As can be seen, the ***RT-Network*** achieved about 58% and ***RT-Binary*** about 55%, there is an improvement of up to 3%. We also utilize the Precision at k - **P@*k*** and ***R-precision*** from Information Retrieval domain, to evaluate $\mathcal{R}a\mathcal{R}^{\varphi}_{v_j}$. The **P@*k*** is the proportion of relevant documents in the first k positions. This leads to measuring precision at fixed low levels of retrieved results, such as 10 or 30 documents. The ***R-precision*** is the precision after R documents have been retrieved, where R is the number of relevant documents [12]. In our study we considered R = 212 and k =10. The ***RT-Network*** achieved ***212-precision*** = 56% and ***RT-Binary*** achieved ***212-precision*** = 50%, ***RT-Network*** attained a little better performance than ***RT-Binary.*** On the other hand, these two networks achieved ***P@10*** = 90%. The best-fit $\mathcal{RR}^{\varphi}_{v_j}$ Equations are illustrated in Table 1. The Table 2 shows the baseline results. We compute the three baseline measures (M_i) for both, ***RT-Network*** and ***RT-Binary***. One can verify that, all three metrics failed to infer user reputation comparing with our approach (see Table 2).

Table 1. The best outcomes achieved

Network	$\mathcal{R}a\mathcal{R}^{\varphi}_{v_j}$	AvP	*212-precision*	P@10
RT-Network	$\mathcal{R}a\mathcal{R}^{1}_{v_j} = \frac{((0.7*Cc)+(0.2*D_{in})+(0.1*D_{out}))}{Max\{(m_i*w_i),..,(m_n*w_n)\}}$ (3)	**58%**	**56%**	**90%**
RT-Binary	$\mathcal{R}a\mathcal{R}^{2}_{v_j} = \frac{((0.3*Prank)+(0.3*D_{in})+(0.1*D_{out})+(0.3*Ec))}{Max\{(m_i*w_i),..,(m_n*w_n)\}}$ (4)	**55%**	**50%**	**90%**

Table 2. The main results of baseline case study

M_i	Baseline Measures	*RT-Network* AvP($\mathcal{R}a\mathcal{R}^{\varphi}_{v_j}$)	*RT-Binary* AvP($\mathcal{R}a\mathcal{R}^{\varphi}_{v_j}$)
M_1	Follower count	18 %	18 %
M_2	PageRank	37 %	45 %
M_3	InDegree	48 %	44 %

Table 3. The ***p@k*** and ***212-precision*** of ***RT-Network*** and ***RT-Binary***

M_i	Baseline Measures	*RT-Network*		*RT-Binary*	
		P@10	*212-precision*	*P@10*	*212-precision*
M_1	Follower count	14%	19%	14%	19%
M_2	PageRank	90%	37%	100%	24%
M_3	InDegree	89%	37%	89%	24%

The Table 3 shows the performance of the baselines rank methodology. The ***212-precision*** of baseline measures is smaller than our rank reputation approach $\mathcal{RaR}^{1}_{v_j}$ and $\mathcal{RaR}^{2}_{v_j}$. We calculate $\mathcal{RaR}^{\varphi}_{v_j}$ using the set of basic measures (Bc, Cc, Dc, Ec) in isolation, for both networks and the results are shown in Table 4. The $\mathbf{AvP}(\mathcal{RaR}^{\varphi}_{v_j})$ varies from 18 to 45%. In all baseline cases, the values of 212-Precision achieved maximum of 50% (Dc measure - Table 5 and RT-Binary - Table 3), and minimum of 14%. The P@10 lies in the range of 0 and 100%. If we considered only AvP performance measure, the metrics in isolation failed to infer user reputation (by comparing with our approach; see Table 1). The rank approach using centralities metrics (Table 4) in isolation performed slightly worse than the baselines metrics M1, M2 and M3 (Table 3). The worst performing approach was Follower count metric (M1), followed by Cc, in both RT-Network and RT-Binary.

Table 4. Results of measures performed in isolation

Measures	*RT-Network*			*RT-Binary*		
	AvP	***P@10***	***212-precision***	**AvP**	***P@10***	***212-precision***
Dc	43%	80%	50%	44%	89%	49%
Cc	31%	80%	33%	18%	0%	14%
Bc	22%	100%	25%	22%	100%	30%
Ec	45%	100%	43%	32%	100%	30%

5 Related Works

Many concerns arise about the data content quality in health domain, and the possibility that poor information is detrimental on health [14, 15]. Others studies use metadata [16, 17]. Although these studies aimed to analyze mostly websites or homepages a little or nothing is available in UCG systems in health domain. There have been a few recent studies on analyzing data in Twitter; Kwak and colleagues [5] rank Twitter users' by popularity. The popularity was estimated by followers count, PageRank and retweet count. All results shown that celebrities (actors, musicians, politicians, sports stars, etc.) or news media were ranked on the top of the list. The PageRank and follower count metrics rank mostly celebrities on the top. The retweet count metrics ranked not only celebrities but also news and media on the top of the list. Cha et al [13] presented an empirical analysis of influence patterns in Twitter. They compared three different measures of influence: indegree, retweets, and mentions. The authors found that, the most influential users were: news sources (CNN, New York Times), politicians (Barack Obama), athletes (Shaquille O'Neal), as well as celebrities (Ashton Kutcher, Britney Spears). The most retweeted users were content aggregation services (Mashable, TwitterTips, TweetMeme), businessmen (Guy Kawasaki), and news sites (The New York Times, The Onion). Finally, the most mentioned users were mostly celebrities. Others measures are also used to rank node importance, such as co-follower rate (ratio between follower and following), frequency of tweets (updates), who your followers follow, etc.

6 Conclusion

This paper explores the rich data structure of social media systems. We exploited the SNA to figure out user's reputation. In our study, reputation has the same meaning of credible source information. Social network creates trust between agents since they allow their members to learn about each other through repeated interactions. In that case, the interaction takes place through retweeting function. We consider that, Twitter's communicative structure is determined by two overlapping and interdependent networks – one based on follower-following relationships, the most obviously; and other relatively short-term and emergent, based on shared interest in a topic or event, often coordinated by a retweet function. Therefore, Retweet Network must be understood as separate from follower/following Network. We found out some interesting results. The majority of Twitter accounts are individual or blogs, since this is the nature of Web 2.0. Web 2.0 is driven by participation and collaboration among users, most obviously apparent in social networking, social bookmarking, blogging, wikis etc. We also found out that Reputation Rank Approach is responsive to Dcin and Dcout. These metrics is present in all best-performing results of RT-Network. On the other hand, Ec and PageRank metrics are present in all best-performing results of RT-Binary. By contrast, the average precision RT-Network and RT-Binary worst-performing results are those that use the PageRank and Bc. All metrics in isolation failed in reach user reputation, specially the Cc and follower count metrics. Almost all measure of RaR achieved about 90% of P@10 performance measure. The study gives us a clear understanding of how measure selection can affect the reputation rank. Choosing the most appropriate measure depends on what we want to represent. We observed that popularity (or key position in a graph) does not necessarily refer to reputation. The Bc metric is an important quantity to characterize how influential a node (user) is in communications between each pair of vertices, it represents a gate between groups node, and yet both metrics failed. By contrast, the Cc and Dcin metrics fulfilled the rank goal, i.e., in expressing the reputation. The major contribution of this work is providing a new methodology to rank trustworthy source using a new network structure based on retweet ties. The result shown that our rank methodology and the network topology model have successfully evaluated user reputation. Since the results indicate that our ranking approach outperformed the baseline case. Additionally, we verified that in Twitter community trust plays an important role in spreading information; the culture of "retweeting" demonstrates the potential to reach trust. It must be stressed that, we find out small values of Dc associated to large Bc, as well, weak correlation between them. These results provided evidences that RT-network maybe incorporates fractal properties. For future work, we will be conduct an in-depth study of fractal properties in order to figure out if it follows others fractal properties, e.g., self-similarity.

References

1. de Boer, M.J., Versteegen, G.J., van Wijhe, M.: 'Patients' use of the Internet for pain-related medical information. Patient Education and Counseling 68(1), 86–97 (2007)
2. Anderson, J.G.: Consumers of e-Health: Patterns of Use and Barriers. Social Science Computer Review 22(2), 242–248 (2004)

3. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In: Proceedings of the International Conference on Web Search and Web Data Mining, Palo Alto, California, USA (2008)
4. Wasserman, S., Faust, K.: Social network analysis. University Press (1994)
5. Kwak, H., Lee, C., Park, H., Moon, S.: What is Twitter, a social network or a news media? In: Book What is Twitter, a Social Network or a News Media?, pp. 591–600. ACM (2010)
6. Cheung, C.M.K., Lee, M.K.O.: A theoretical model of intentional social action in online social networks. Decision Support Systems 49(1), 24–30 (2010)
7. Cha, M., Mislove, A., Gummadi, K.P.: A measurement-driven analysis of information propagation in the flickr social network. In: Proceedings of the 18th International Conference on World Wide Web, Madrid, Spain (2009)
8. Shen, Y., Syu, Y.S., Nguyen, D.T., Thai, M.T.: Maximizing circle of trust in online social networks. In: Book Maximizing Circle of Trust in Online Social Networks, pp. 155–164 (2012)
9. Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.: The role of social networks in information diffusion. In: Book The Role of Social Networks in Information Diffusion, pp. 519–528 (2012)
10. Bonacich, P.: Some unique properties of eigenvector centrality. Social Networks 29(4), 555–564 (2007)
11. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. In: Book The PageRank Citation Ranking: Bringing Order to the Web, Stanford InfoLab (1999)
12. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc. (1999)
13. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.: Measuring User Influence in Twitter: The Million Follower Fallacy. In: Book Measuring User Influence in Twitter: The Million Follower Fallacy. AAAI Press, Menlo Park (2010)
14. Weitzel, L., Quaresma, P., de Oliveira, J.P.M.: Evaluating Quality of Health Information Sources. In: Book Evaluating Quality of Health Information Sources, pp. 655–662. IEExplore (2012)
15. Stvilia, B., Mon, L., Yi, Y.J.: A model for online consumer health information quality. Journal of the American Society for Information Science and Technology 60(9), 1781–1791 (2009)
16. Shon, J., Musen, M.A.: The low availability of metadata elements for evaluating the quality of medical information on the World Wide Web. In: Proceedings/AMIA. Annual Symposium. AMIA Symposium, pp. 945–949 (1999)
17. Wang, Y., Liu, Z.: Automatic detecting indicators for quality of health information on the Web. International Journal of Medical Informatics 76(8), 575–582 (2007)

Reverse Engineering of Database Security Policies

Salvador Martínez[1], Valerio Cosentino[1], Jordi Cabot[1], and Frédéric Cuppens[2]

[1] ATLANMOD, & École des Mines de Nantes, INRIA, LINA, Nantes, France
{salvador.martinez_perez,valerio.cosentino,jordi.cabot}@inria.fr
[2] Télécom Bretagne; Université Européenne de Bretagne Cesson Sévigné, France
frederic.cuppens@telecom-bretagne.eu

Abstract. Security is a critical concern for any database. Therefore, database systems provide a wide range of mechanisms to enforce security constraints. These mechanisms can be used to implement part of the security policies requested of an organization. Nevertheless, security requirements are not static, and thus, implemented policies must be changed and reviewed. As a first step, this requires to discover the actual security constraints being enforced by the database and to represent them at an appropriate abstraction level to enable their understanding and reenginering by security experts. Unfortunately, despite the existence of a number of techniques for database reverse engineering, security aspects are ignored during the process. This paper aims to cover this gap by presenting a security metamodel and reverse engineering process that helps security experts to visualize and manipulate security policies in a vendor-independent manner.

1 Introduction

Relational databases are at the core of most companies information systems, hosting critical information for the day to day operation of the company. Securing this information is therefore a critical concern. For this purpose, both, researchers and tool vendors have proposed and developed security mechanisms to ensure the data within the database system is safe from possible threats. Due to its relative conceptual simplicity, one of the most used mechanisms within DataBase Management Systems (DBMSs) are *access control (AC)* policies where Role-based access control (RBAC) [10] is the current trend.

However, and despite the few methods that attempt to automatically derive these policy implementations from high-level security specifications[8,3,2], the task of implementing an access control security policy remains in the vast majority of cases a manual process which is time-consuming and error-prone. Besides, several database mechanisms may be needed in combination to implement a policy, e.g., triggers can be used to add fine-grained control on privileges, scattering the policy and increasing the complexity of the definition process. Furthermore, as security requirements are rarely static, frequent modifications of the security policy implementation are required, what increases the chances of introducing new errors and inconsistencies.

In this context, discovering and understanding which security policies are actually being enforced by the DBMS comes out as a critical requirement for the reenginering

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 442–449, 2013.

of the current policies, to adapt them to evolving needs of the company and also to detect inconsistencies between the enforced and the desired ones. The main challenge for this discovery process is bridging the gap between the vendor-dependent policy representation and a more logical model that 1 - express these policies in a way that abstracts them from the specificities of particular database systems and 2 - that can be understood by security experts with no deep knowledge of the particularities of each vendor. Representing and processing the security policies at this logical level is much easier than a direct exploration of the database dictionary whose structure is unfortunately not standardised. Additionally, this logical model would also allow us to implement all analysis/evolution/refactoring operations on the security policies in a vendor-independent and reusable way.

The goal of this paper is then two-fold. First we provide a means to represent such logical models for security concerns in relational DBMSs. Secondly, we describe a reverse engineering approach that can automatically create this logical model out of an existing database. Reverse engineering, as a process aimed to represent running systems at higher abstraction level, has been proved useful in many domains, including database systems [7], [4]. However, these works have focused on the database structure and ignored the security aspects. We intend to cover this gap.

Moreover, we discuss possible applications and benefits of using a model-based representation given this enables to reuse in the security domain the large number of model-driven techniques and tools. Model manipulation techniques can then be applied to visualize, analyze, evolve, etc the model. Then, a forward engineering process could be launched in order to generate the new security policy implementation ready to import in the target database system.

The rest of the paper is organized as follow. In Section 2 a metamodel able to represent models of database AC policies is provided whereas in 3 we show the needed steps to obtain such models from an existing database. In 4 we discuss the related work. We finish the paper with section 5 presenting the conclusions and future works.

2 Access Control Metamodel

This section describes our relational database-specific access control metamodel (RBAC-inspired). Next section describes how to automatically populate it based on the actual policies enforced in a particular database.

The SQL standard predefines a set of privileges and object types that can be used in the definition of a relational database. Our metamodel provides a direct representation of these concepts to facilitate the understanding of the security policies linking the security elements with the schema objects constrainted by them as depicted in figure 1. In the following, we describe the main elements of the metamodel.

Database Objects: The main objects users can be granted privileges on are: tables, columns, views, procedures and triggers. Each one of these elements is represented in the metamodel by a metaclass inheriting from the *SchemaObject* metaclass. Views (metaclass *View*) can include derived columns. In fact, views can be used as a fine-grained access control mechanism. A view filters table rows and columns with respect

to a desired criteria so when a subject is granted privileges over a view, is actually being granted privileges over a table/s but with certain constraints. This actually represents *column-level* and *content-based (row level)* access control.

Privileges: We can divide the privileges that can be granted in a DBMSs in five categories: *Database-level* privileges, for those privileges that imply creation of database objects including users and roles. *Table-level* privileges for those that implies table, columns and index access. *Permission-delegation* privileges to delegate permission administration to users and roles. *Execute* privileges for the executable elements (stored procedures and functions) and *Session* privileges. These categorization is depicted in the bottom part of Figure 1. If needed, an extension of this metamodel (by inheriting from the *Operation* metaclass) could be provided to deal with vendor-specific privileges.

Subjects: Subjects executing actions on the DBMS are users and roles and they are, as such, represented in the metamodel with the corresponding metaclasses *User* and *Role*, possibly part of role hierarchies. In the metamodel, the metaclass *Role* inherits from the metaclass *Subject* which enables it to become grantee. The *User* metaclass also inherits from *Subject* what means that, privileges, in contrast to pure RBAC, can be granted to both users and roles.

Other Elements and Constraints: There are several other aspects that have to be taken into account. The execution of privileges may need to be constrained. In DBMSs this is normally achieved by using triggers and procedural code. The metamodel should permit the representation of the existence of such constraints. The metaclass *Constraint* meets that purpose. Typically it points to a *Trigger* linked to the object and the operation on it that is constrained by the trigger. The *Trigger* metaclass also includes the attributes the trigger body and the condition and error message to be displayed when the trigger execution fails due to any exception. Another aspect to be considered is object ownership as it is the basis for permission delegation. An association between objects and subjects records this relationship. Finally, global permission, i.e., permissions that are granted on all the elements of the same type are represented by allowing in the metamodel permissions to be granted on any object, including the metaclasses *Schema* and *Database*. A select permission granted on *Schema* or *Database* represents that the grantee can select all the tables and views contained in those objects.

3 Reverse Engineering Process

The previous metamodel allows to represent database access control policies at the logical level. Models conforming to this metamodel express the security policies in place in a given database in a vendor-independent manner. These models can be manually created but ideally they should be automatically created as part of an automatic reverse engineering process that instantiates the model elements based on the information extracted out of the DBMSs. Note that, the extracted models are vendor-independent but the extraction process is not since each vendor uses different internal structures (i.e. a different set of tables/columns in the data dictionary to express this information). In fact, we could regard this “injection” process as the one that abstracts out the specific product details.

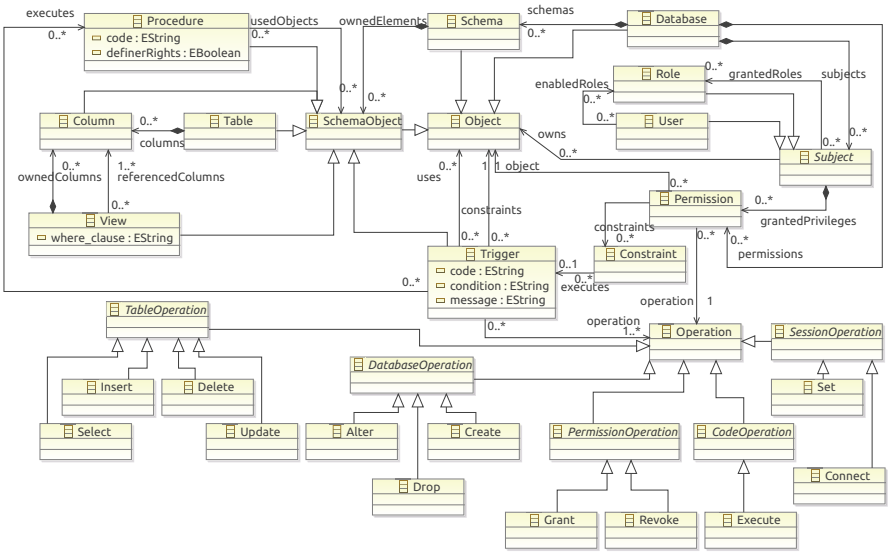


Fig. 1. DBMS Security Metamodel

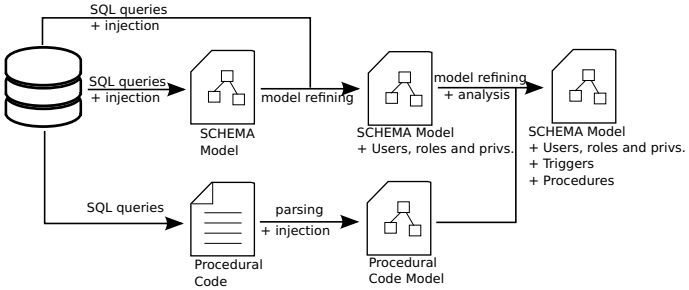


Fig. 2. Extraction process

Our reverse-engineering approach is depicted in Figure 2. It starts by populating our security model with the basic schema information (tables, views, etc). Then this model is refined to add user, roles and privileges information whereas in a last step, the bodies of triggers and stored procedures are analyzed to complement the access control policies in the model.

In the following we will detail the process for the reverse engineering of security policies over an Oracle 10g DBMS. This same process could be reused for other DBMSs changing the references to the specific data dictionary tables with the corresponding ones in that system.

3.1 Extracting General Schema Information

This first step populates the part of the model describing the structure of the database itself e.g., schemas, tables (and columns), views, procedures, etc. An injector (in our

terminology, an injector is a software component that bridges technical spaces, in this case moving information from the database technical space to the modeling technical space) is needed. This injector connects to the database and queries the dictionary tables to retrieve all the necessary information. The selected objects are then inserted as model elements in the security model. In Oracle, our injector has to query tables like *DBA_TABLES*,*DBA_TAB_COLS* and *DBA_ROLES*.
View extraction is more challenging since we need to parse the view definition to get the list of tables (or other views), columns and conditions the view selects. The result of the parsing is a set of links between the view object and the other objects filtered by the view. The *where* clause will be copied as it is into the *condition* attribute of the view metaclass. Moreover, a view can contain derived columns. Derived columns are created as view columns in the model.

3.2 Extracting Users, Roles and Privileges

As before, the injector queries the database dictionary for user and roles information and populate the model with the results. After this step, the general access control information of the database, i.e., subjects, objects and permissions are already present in the security model.

3.3 Extracting Stored Procedures and Triggers Information

Triggers. Complex security checks can be implemented by means of triggers that fire to prevent certain actions when performed in a certain context. However, triggers are used for a huge variety of tasks beyond security purposes. In our approach, all triggers are retrieved and parsed, then, they are analyzed with respect to a number of heuristic conditions in order to select which of them are implementing security checks and discard the rest.

The heuristic conditions analyze the following aspects:

-Trigger kind: Triggers are fired before/after a certain statement/s is invoked. *BEFORE STATEMENT* triggers are executed before the statement is completed, enabling the possibility of evaluating security concerns (e.g., the possibility to make inserts in certain tables could be enabled only to working days). Conversely, *AFTER STATEMENT* triggers are executed once the action of the statement is performed, so, when involved in security, they are normally used for logging purposes. Clearly, our focus should be in the *BEFORE STATEMENT* triggers. *-Trigger contents:* Although the kind of the trigger is an important hint, it is not enough. We analyze the trigger actions and conditions to find operations that are likely to be used when performing security checks like system context information checks, user information checks, exception raising's, etc.

More specifically, a trigger will qualify as a *security trigger* if it fulfills all the following heuristic conditions:

1. The trigger is a before statement trigger.
2. The trigger contains an exception section that raises an exception.
3. The trigger evaluates conditions on the system (ip address, host, time) or user information (name, assigned privileges).

Listing 1.1. Trigger's code

```
TRIGGER update_job_history
  AFTER UPDATE OF job_id,
       department_id ON employees
  FOR EACH ROW
BEGIN
  add_job_history(:old.employee_id, :
       old.hire_date, sysdate,
    :old.job_id, :old.department_id);
END;

TRIGGER secure_employees
  BEFORE INSERT OR UPDATE OR DELETE
       ON employees
BEGIN
  IF TO_CHAR (SYSDATE, 'HH24:MI') NOT
        BETWEEN '08:00' AND '18:00'
        OR TO_CHAR (SYSDATE, 'DY') IN
             ('SAT', 'SUN') THEN
     RAISE_APPLICATION_ERROR (-20205,
    'You may only make changes during
          normal office hours');
  END IF;
END secure_employees;
```

Listing 1.2. Procedure code

```
PROCEDURE add_job_history
 (
     p_emp_id job_history.employee_id
          %type
   , p_start_date job_history.
        start_date%type
   , p_end_date job_history.end_date%
        type
   , p_job_id job_history.job_id%type
   , p_department_id job_history.
        department_id%type
   )

IS
BEGIN
  INSERT INTO job_history
    (employee_id, start_date,
         end_date,
      job_id, department_id)
    VALUES(p_emp_id, p_start_date,
           p_end_date, p_job_id,
           p_department_id);
END add_job_history;
```

As an example, listing 1.1 shows two triggers from the well-know Human Resources (HR)[1] schema example privided by Oracle: *Secure_employees* and *Update_job_history*. The latter one is an *AFTER STATEMENT* trigger which directly disqualifies it as a security enforcement trigger. Conversely, the former fulfils all the heuristic conditions. It is a *BEFORE STATEMENT* trigger, it raises and exception and checks system information (the time) and thus it qualifies as a *security trigger*.

Once a trigger is identified as a *security trigger*, the constraints imposed by the trigger need to be extracted and added to the model.

Stored Procedures. Stored procedures can be executed with invoker or definer's rights. In the latter, the invoker role gets transitive access to certain database objects. The source code of the procedure must be analysed to obtain such set of transitive permissions. Our approach parses the store procedures and extracts the accessed database objects. These are then linked to the procedure in the database security model in order to easily retrieve them later on during the analysis phase. As an example, the *add_job_history* procedure in listing 1.2 is declared to be invoked with definer rights. The table it accesses, *Job_history*, would appear linked to this procedure in the model.

3.4 Operations

The generated security model can be used in many different scenarios such as visualizations, queries, metrics, refactoring and reengineering operations. Note that working at the model level we can benefit from all existing model-driven techniques when implementing these scenarios and, furthermore, we can work at a vendor-independent level which means they can be used no matter the database system in place.

[1] `http://docs.oracle.com/cd/B13789_01/server.101/b10771/scripts003.htm`

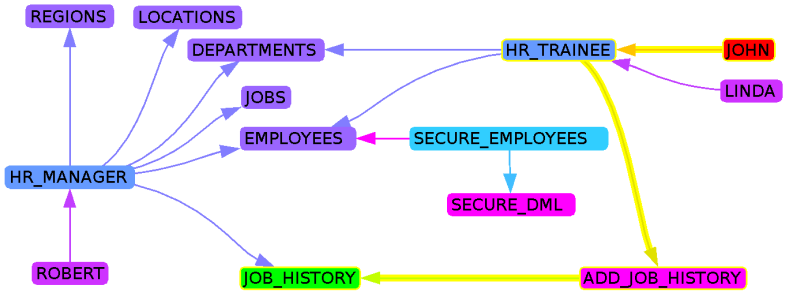


Fig. 3. Database security model visualization

As an example of a possible interesting scenario, we show in Figure 3 a visualization obtained from a model extracted out of a modification of the HR schema. where each kind of element and relation is shown is a different color. This visualization helps to quickly grasp important information. Moreover, we can easily check if a given user can access a given table or view by just checking existing paths as the one highlighted between the user *JOHN* and the table *JOB_HISTORY*.

4 Related Work

To the best of our knowledge, ours is the first security metamodel tailored to the security mechanisms available in relational databases. Our metamodel integrates RBAC[10] concepts but extends the standard RBAC model to cover the full spectrum of database security mechanisms useful to express access-control policies. Although several (extensions to) modelling languages able to model security concerns[5,6] have already been proposed, they are aimed to model the security aspects of the whole information system and thus lack of precision to define in detail database-specific access control policies.

Regarding the reverse-engineering of security aspects, in [11] the authors present an approach to discover and resolve anomalies in MySQL access-control policies by extracting the access-control rules and representing them in the form of Binary Decision Diagrams. They do not provide a higher-level, easier to understand and manipulate representation of the extracted policies nor take into account the contribution that several other elements like triggers, stored procedures and views provide to access-control.

Finally, there exist a plethora of reverse engineering efforts [7], [4], [9],[1] (among many others) focused in recovering a (conceptual or logical) schema from a database. Nevertheless, none of them covers security aspects and therefore, they could benefit from our approach to extract a richer model.

5 Conclusions and Future Work

We have presented a reverse engineering process to extract a logical model of the security constraints enforced by a database. This model is vendor-independent and, as

such, facilitates the analysis of the implemented policies by security experts not necessarily familiar with the specific details of each database system. This also facilitates comparing the security policies across different units of the company to make sure their policies are consistent with each other.

As future work we plan to continue working on the applications mentioned in section 3 and investigate the benefits of raising further the abstraction level by generating XACML specifications from our logical model to benefit from existing general security algorithms during the analysis of the security policies. Moreover, we would like to exploit the information of database audits to check the actual usage of the implemented policies (e.g. which permissions are more frequently used, which roles are never employed,...) and/or attempted attacks.

References

1. Astrova, I.: Towards the semantic web - an approach to reverse engineering of relational databases to ontologies. In: ADBIS Research Communications. CEUR Workshop Proceedings, vol. 152 (2005), CEUR-WS.org
2. Barker, S., Douglas, P.: RBAC policy implementation for SQL databases. In: DBSec, pp. 288–301 (2003)
3. Basin, D., Doser, J., Lodderstedt, T.: Model driven security: From UML models to access control infrastructures. ACM Trans. Softw. Eng. Methodol. 15, 39–91 (2006)
4. Chiang, R.H.L., Barron, T.M., Storey, V.C.: Reverse engineering of relational databases: extraction of an EER model from a relational database. Data Knowl. Eng. 12, 107–142 (1994)
5. Jürjens, J.: UMLsec: Extending UML for secure systems development. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 412–425. Springer, Heidelberg (2002)
6. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-based modeling language for model-driven security. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 426–441. Springer, Heidelberg (2002)
7. Iuc Hainaut, J., Chandelon, M., Ch, M., Tonneau, C., Joris, M.: Contribution to a theory of database reverse engineering. In: Proc. of the IEEE Working Conf. on Reverse Engineering, pp. 161–170. IEEE Computer Society (1993)
8. Oh, S., Park, S.: Enterprise model as a basis of administration on role-based access control. In: CODAS 2001, pp. 165–174 (2001)
9. Petit, J.-M., Kouloumdjian, J., Boulicaut, J.-F., Toumani, F.: Using queries to improve database reverse engineering. In: Loucopoulos, P. (ed.) ER 1994. LNCS, vol. 881, pp. 369–386. Springer, Heidelberg (1994)
10. Sandhu, R., Ferraiolo, D., Kuhn, R.: The NIST model for role-based access control: towards a unified standard. In: Proceedings of the Fifth ACM Workshop on Role-Based Access Control, RBAC 2000, pp. 47–63. ACM (2000)
11. Shehab, M., Al-Haj, S., Bhagurkar, S., Al-Shaer, E.: Anomaly discovery and resolution in mySQL access control policies. In: Liddle, S.W., Schewe, K.-D., Tjoa, A.M., Zhou, X. (eds.) DEXA 2012, Part II. LNCS, vol. 7447, pp. 514–522. Springer, Heidelberg (2012)

Discovering Multi-stage Attacks Using Closed Multi-dimensional Sequential Pattern Mining

Hanen Brahmi[1] and Sadok Ben Yahia[2]

[1] Faculty of Sciences of Tunis
Computer Science Department
Campus University, 1060 Tunis, Tunisia
hanenbrahmi@gmail.com
[2] Institut Mines-TELECOM, TELECOM SudParis,
UMR CNRS Samovar, 91011 Evry Cedex, France
sadok.benyahia@fst.rnu.tn

Abstract. Due to the growing amount and kinds of intrusions, multi-stage attack is becoming the one of the main methods of the network security threaten. Although, the Intrusion Detection Systems (IDS) are intended to protect information systems against intrusions. Nevertheless, they can only discover single-step attacks but not complicated multi-stage attacks. Consequently, IDS are plugged with the problem of the excessive generation of alerts. Therefore, it is not only important, but also challenging for security managers to correlate security alerts to predict a multi-stage attack. In this respect, an approach based on sequential pattern mining technique to discover multi-stage attack activity is efficient to reduce the labor to construct pattern rules. In this paper, we introduce a novel approach of alert correlation, based on a new closed multi-dimensional sequential patterns mining algorithm. The main idea behind this approach is to discover temporal patterns of intrusions which reveal behaviors of attacks using alerts generated by IDS. Our experiment results show the robustness and efciency of our new algorithm against those in the literature.

Keywords: Multi-stage attacks, Intrusion detection system, Multi-dimensional sequential patterns, Alert correlation.

1 Introduction

Nowadays, as a security infrastructure the *Intrusion Detection System* (IDS) have evolved significantly since their inception.

The criticism of the weakness of IDSs focuses on the fact that it can be violated by a multi-stage attack. The latter is a complex attack that comprises multiple correlated intrusion activities. A single multi-stage attack may generate a huge amount of uncorrelated intrusion alerts. In order to cope with such quantities of alerts, *alert correlation* approaches are used [1,2,5]. Alert correlation is the task of analyzing the alerts triggered by one or multiple IDSs. Suppose a security analyst is responsible for analyzing a huge number of this kind of alerts and

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 450–457, 2013.

to take appropriate decisions. It is clear that it will be embedded in the huge amount of knowledge and be found quickly overwhelmed. Therefore, the critical task of analyzing and correlating all these alerts is time consuming and prone to human errors.

The input data for alert correlation tools can be gathered from various sources such as IDSs, fire-walls, web server logs, *etc.*. All the alerts stored in databases can be viewed as sequences of alerts related to a time order. In this respect, it has been shown that finding out multi-stage attacks by analyzing the alert sequences is a promising method [4]. Alongside, the set of alerts can be considered as a multi-dimensional data stored in a *Data Warehouses* (DW). In this respect, alert correlation techniques using DW will have several advantages [1]. In fact, it will provide a high level representation of the alerts along with a temporal relationship of the sequence in which these alerts occurred.

In this paper, we investigate another way aiming at discovering the multi-stage attacks based on a DW perspective. Thus, we introduce a new alert correlation system, which focuses on the multi-dimensional and temporal characteristics of alerts. The system is based on a closed multi-dimensional sequential pattern mining algorithm, called *Closed Multi-Dimensional Prefix Span* (CMD_PrefixSpan). Through extensive carried out experiments on the real-life log of alerts flagged out by IDS Snort[1], we show the significance of our approach in comparison with those fitting in the same trend.

The remaining of the paper is organized as follows. Section 2 sheds light on some related work. We introduce our novel approach to discover multi-stage attacks in Section 3. We also relate the encouraging results of the carried out experiments in Section 4. Finally, Section 5 concludes and points out avenues of future work.

2 Related Work

In order to discover composite multi-stage attacks, several approaches for alert correlation have been proposed.

Indeed, Ning et *al.* [5], Cuppens and Miège [2] built alert correlation systems based on matching the pre-/post-conditions of alerts. The proposed methods originate from the idea that attack steps are directly related, since an earlier attack enables or positively affects the later one. One challenge of these approaches is that a new attack cannot be paired with any other attacks, since its prerequisites and consequences are not defined.

Along with the same preoccupation, Li et *al.* [4] proposed an approach based on sequential pattern mining technique to discover multi-stage attack patterns. The authors apply the GSP algorithm [8], to mine attack behavior sequence patterns from alarm databases.The main moan that can be addressed to this approach stands in the fact that it inherits the drawbacks of GSP algorithm. Hence, this approach may not be sufficient in mining large sequence databases having numerous long patterns as the alert logs.

[1] Snort is a free open source IDS available at `http://www.snort.org/`.

The main thrust of this paper is to propose a new algorithm for mining multi-dimensional closed sequential patterns, called CMD_PrefixSpan, to discover multi-stage attack behavior patterns. The main reason behind our choice comes from the observation that a multi-stage attack usually has relatively fixed attack pattern and occurs in a confined time span [4]. Moreover, it has been shown that the alerts have a multi-dimensional characteristic [1]

3 Discovering Multi-stage Attacks

The alerts can be stored as a global sequence of alerts sorted by ascending time of detection (timestamp). The sequences of alerts describe the behaviors and actions of attackers. Thus, we can find out multi-stage attacks by analyzing these alert sequences.

We present in the following the key settings that will be of use in the remainder.

3.1 Formal Background

Let a set of records be a global sequence of alerts. The set is denoted $\mathcal{SA}$ = <A-id, S>, where A-id is an alert identifier and S is a set of alert information sequences occurring in the same time. Our algorithm only focuses on the attack names and timestamp attributes in the alerts.

Definition 1. ***(n-dimensional sequence)****. A n-dimensional sequence (n > 1) is an ordered list of (n-1)-dimensional sequence. It can be denoted as* $<s_1 s_2 ... s_r>_n$*, where n is the number of dimensions and* s_i *is a (n-1)-dimensional sequence element for* $1 \leq i \leq r$*.*

The support of s_i *is the number of sequences containing* s_i *in* $\mathcal{SA}$*, denoted as sup(*s_i*). Given a user-specified minimum support threshold min_Supp, the set of frequent multi-dimensional sequential patterns, denoted by* $\mathcal{FMDS}$*, includes all multi-dimensional sequences whose support is greater than or equal to the min_Supp.*

Example 1. Table 1 shows an example of a global sequence of alerts, with its time of detection timestamp in ascending order. Table 2 illustrates the multiple subsequences, $\mathcal{SA}$, generated by dividing the global alert sequence according to timestamp and the name of attack. Note that, in order to run quickly, we map the attack names to letter.

According to $\mathcal{SA}$ shown in Table 2, the subscripts in alert information sequences denotes the dimension numbers. The dimensions 1, 2 and 3 represent the attack names, the sessions and the days, respectively. <<<ab>$_1$<dc>$_1$>$_2$<<df>$_1$ <bc>$_1$>$_2$>$_3$ is a 3-dimensional sequence, whose elements <<ab>$_1$<dc>$_1$>$_2$ and <<df>$_1$<bc>$_1$>$_2$ are 2-dimensional sequences.

Given $min\_Supp = 2$, the 1-dimensional sequence element <ab>$_1$ is a frequent multi-dimensional sequential pattern in $\mathcal{SA}$. In fact, there are two alert records A_1 and A_2 which support the pattern in $\mathcal{SA}$.

Table 1. A global sequence of alerts

Attack Names	Timestamp	Other attributes
...	...	...
a	11-08-09-16:12:14	...
b	11-08-09-16:13:18	...
c	11-08-09-16:34:46	...
d	11-08-09-16:45:10	...
f	11-08-10-02:06:53	...
c	11-08-10-02:12:47	...
...	...	...

Table 2. A multi-dimensional sequence of alerts $\mathcal{SA}$

A-id	Alert information'
A_1	$<<<ab>_1<dc>_1>_2<<df>_1<bc>_1>_2>_3$
A_2	$<<<ab>_1>_2<<c>_1<be>_1>_2<<dg>_1<hf>_1>_2>_3$
A_3	$<<<bc>_1<ac>_1>_2<<fh>_1<ab>_1>_2>_3$

Definition 2. *(**l-sequence**). The total number of occurrences of patterns in a sequence s is called the **length** of the sequence. An n-dimensional sequence with a length l, is called l-sequence.*

Example 2. A= $<<<ab>_1<dcf>_1>_2<<df>_1<bc>_1>_2>_3$ is a 9-sequence, since pattern a occurs 1 time, pattern b 2 times, pattern c 2 times, pattern d 2 times and pattern f 2 times in the sequence. Consequently, we have A(1)=a, means that the 1st occurrence in A is a.

According to the multi-dimensional sequence of alerts $\mathcal{SA}$ shown in Table 2, there are two different representations of $\mathcal{SA}$: the *standard* and the *simplified* formats as shown in Table 3.

Table 3. $\mathcal{SA}$ in both standard and simplified formats

A-id	$\mathcal{SA}$ in standard format	$\mathcal{SA}$ in simplified format
A_1	$<<<ab>_1<dc>_1>_2<<df>_1<bc>_1>_2>_3$	$<$(a1) (b1) (d2) (c1) (d3) (f1) (b2) (c1)$>_3$
A_2	$<<<ab>_1>_2<<c>_1<be>_1>_2<<dg>_1<hf>_1>_2>_3$	$<$(a1) (b1) (c3) (b2) (e1) (d3) (g1) (h2) (f1)$>_3$
A_3	$<<<bc>_1<ac>_1>_2<<fh>_1<ab>_1>_2>_3$	$<$(b1) (c1) (a2) (c1) (f3) (h1) (a2) (b1)$>_3$

The standard format of $\mathcal{SA}$ can be transformed into the simplified format. The conversion is based on the definition of the dimensional scope relation DS(S, k, j) defined as follows.

Definition 3. *(**Dimensional Scope relation**). Given an n-dimensional alert information sequence S that includes two patterns S(k) and S(j), such as k and j denote the i-th occurrence of patterns in S. Let s be an i-dimensional sequence element of S that contains S(k) and S(j) in its different element. Then, the dimensional scope of S(k) and S(j) is DS(S, k, j) = i.*

Example 3. Let A = $<<<ab>_1<dcf>_1>_2<<df>_1<bc>_1>_2>_3$ be a 3-dimensional sequence. DS(A, 1, 2) = 1, DS(A, 2, 3) = 2, DS(A, 3, 4) = 1, DS(A, 4, 5) = 1.

Definition 4. *(**Prefix**). Given two n-dimensional sequences $A = <s_1\ s_2 ... s_p>_n$ and $B = <f_1\ f_2\ ...\ f_q>_n$ $(q \leq p)$, B is a prefix of A if and only if $s_i = f_i$ for 1 $\leq i \leq q$.*

Example 4. Let A = <<<ab>$_1$<dcf>$_1$>$_2$<<df>$_1$<bc>$_1$>$_2$>$_3$ be a 3-dimensional sequence, with a simplified format <(a1) (b1) (d2) (c1) (d3) (f1) (b2) (c1)>$_3$. <(a1)>$_3$, <(a1)(b1)>$_3$ and <(a1) (b1) (d2)>$_3$ are the prefixes of A.

Definition 5. ***(Projection)****. Consider two n-dimensional sequences A = <s_1 s_2 ... s_p>$_n$ and B = <f_1 f_2 ... f_q>$_n$ ($q \leq p$) such that B is a prefix of A. A subsequence A' = <g_1 g_2 ... g_r>$_n$ of sequence A is called* a projection *of A with respect to B if and only if (1) A' has prefix B and (2) the last $(p - i_q)$ elements of A' are the same as the last $(p - i_q)$ elements of A.*

Example 5. For instance, if we project A = <(b1) (c1) (a2) (c1) (f3) (h1) (a2) (b1)>$_3$ with respect to prefix B = <(a1)>$_3$, then we have two different projections A', where the first is <(a1) (c1) (f3) (h1) (a2) (b1)>$_3$ and the second <(a1) (b1)>$_3$.

Definition 6. ***(Postfix.)****Let A'=<g_1 g_2 ... g_r>$_n$ be the projection of A = <s_1 s_2 ... s_p>$_n$ with respect to prefix B =<g_1 g_2 ... g_q>$_n$. Then C =<g_{q+1} g_{q+2} ... g_r>$_n$ is called the postfix of A with respect to prefix B. In this respect, we conclude that the postfix can be easily obtained by directly removing the prefix from the projection.*

Example 6. By removing the prefixes from the projections of A = <(b1) (c1) (a2) (c1) (f3) (h1) (a2) (b1)>$_3$, the resulting postfixes of the projection <(a1) (c1) (f3) (h1) (a2) (b1)>$_3$ is <(c1) (f3) (h1) (a2) (b1)>$_3$ and of the second projection <(a1) (b1)>$_3$ is <(b1)>$_3$.

Definition 7. ***(Projected database)****. The α-projected database, denoted by $S|\alpha$, is the collection of postfixes of sequences in database S with respect to α.*

Definition 8. ***(Closed frequent multi-dimensional sequence)****. The closure γ of a frequent multi-dimensional sequential pattern A is the maximal superset of A having the same support value as that of A. Hence, the set of closed frequent multi-dimensional sequential patterns, denoted by $\mathcal{FCMDS}$, includes all sequences in $\mathcal{FMDS}$ which do not have a super-sequence with the same support, such as:*
$\mathcal{FCMDS} = \{A | A \in \mathcal{FMDS} \wedge (\nexists\ B \in \mathcal{FMDS} |\ A \subseteq B \wedge sup(A) = sup(B))\}$.

3.2 The CMD_PrefixSpan Algorithm

Originally, the PrefixSpan algorithm [7] is used for mining frequent sequential patterns in transaction database to discovery the customer purchase patterns. Consequently, it is not straightforward to apply this algorithm to find closed multi-dimensional sequential patterns within alert logs. In fact, PrefixSpan does not consider the fact of having a dimensional scope values between frequent patterns in a projected database. Moreover, it produces an overwhelming large number of sequential patterns. Several of them are considered as redundant. These problems worsen if several dimensions are considered [3,9].

As a solution, we enhance the efficiency of the extraction by considering condensed representations of the mined knowledge, based on the notion of closed patterns [6]. The pseudo-code is shown by Algorithm 1.

Algorithm 1. CMD_PrefixSpan

Input: The alert information' sequences $\mathcal{SA}$, and the minimum support threshold *min_Supp*
Output: The frequent closed set of sequential patterns $\mathcal{FCMDS}$

```
1  Begin
2    // 1st step: Construction of MTable;
3    Foreach s ∈ S|α do
4      for i=1 to Length(s) do
5        p = pattern of n-dimensional sequence element s_i;
6        DS = max(s_1.scop, s_2.scop, ..., s_i.scop);
7        If p is a frequent pattern and cell(p, DS) of MTable has not yet
         counted in the same A-id then
8          add 1 to the cell(p, DS);
9    // 2nd step: Mining of frequent multi-dimensional sequential patterns
10   for i=1 to n do
11     Foreach pattern p found in MTable do
12       If the support count of cell(p, i) ≥ min_Supp then
13         Append p to α to form α' and its dimensional scope given that
           the last pattern of α is i;
14         Store α' in FMDS;
15   Foreach α' ∈ FMDS do
16     Construct α'-projected database;
17     If the size of α'-projected database is larger than min_Supp then
18       Call CMD_PrefixSpan(α', Length+1, S|α');
19   // 3rd step: Derivation of frequent closed multi-dimensional sequential
     patterns
20   FCMDS = ∅;
21   forall α' ∈ FMDS do
22     FCMDS = FCMDS ∪ {γ(α')}
23   return FCMDS
24 End
```

Within the **first step**, the CMD_PrefixSpan algorithm uses a in-memory data structure which is called MTable (*Multi-dimensional Table*) (lines 2−8). MTable allows to store the scope relations, where each column corresponds to a pattern and each row corresponds to a dimensional scope value. Each cell(p, i) in the table records the number of alerts in the projected database S|α that contains pattern p and its dimensional scope knowing that the last element of sequence α is i. After constructing the table, it is easy to find those cells that are frequent. The **second step** of the proposed algorithm uses frequent prefixes to divide the search space and to project sequence databases, as well as the frequent cells founded within MTable (lines 9−18). By recursively finding the sequential patterns in the projected database, we will finally derive all the frequent multi-dimensional sequential patterns from $\mathcal{SA}$. The resulted set is stored in $\mathcal{FMDS}$. Finally, the **third step** allows to compress the latter by finding the closed multi-dimensional

sequential patterns in an alert log (lines 19−22). The obtained $\mathcal{FCMDS}$ exactly captures all the information enclosed in an alert log.

4 Experimental Results

To evaluate the effectiveness and efficiency of our proposed algorithm CMD_PrefixSpan, we carried out extensive experiments on a PC equipped with a 3 GHz Pentium IV and 8 Go of main memory running under Linux Fedora Core 6.

During the carried out experiments, we use a real-life log of alerts flagged out by the IDS Snort deployed on a network of "*National Security Agency*" from "*Inter-Service Academy Cyber Defense Competition*"[2] and captured during the period November 2008 to November 2011.

Figures 1 (a, b, c) plot the comparison of the CMD_PrefixSpan, EXT-Prefixspan and ExtSeq-MIDim algorithms in terms of running times. We take into account three different perspectives, including: (1) different support thresholds; (2) different data volumes; and (3) different number of dimensions.

Basically, the results of Figure 1 (a) shows that the running times of CMD_PrefixSpan, EXT-Prefixspan and ExtSeq-MIDim increase rapidly as the minimum support decreases continuously. Whenever, we continue to decrease the *min_Supp* (*e.g.*, to 0.02%), the CMD_PrefixSpan algorithm will outperform a lot the two frequent multi-dimensional sequence mining algorithms due to the generation of an explosive number of frequent sequences for the latters.

The second simulation is to compare the scalability in terms of running times for different volume of alerts. During this experiment, we fix the *min_Supp* as 0.02%. In this respect, according to Figure 1 (b), we remark that the running time of the three algorithms linearly increases with the number of alerts. Otherwise, it is clear that CMD_PrefixSpan is faster than both EXT-Prefixspan and ExtSeq-MIDim algorithms.

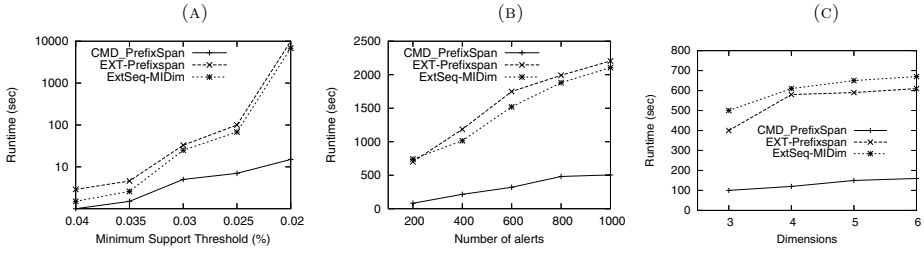


Fig. 1. The running times of CMD_PrefixSpan *vs.* EXT-Prefixspan and ExtSeq-MIDim

In addition, we compare the running times of the algorithms when the number of dimensions changes. To this end, we present Figure 1 (c) where the *min_Supp* is fixed as 0.02%. The results indicate that the CMD_PrefixSpan algorithm

[2] `http://www.itoc.usma.edu/research/dataset/`

performs much better than the EXT-Prefixspan and ExtSeq-MIDim algorithms. The main reason is that the former is not influenced by this factor while the latters deteriorate when the number of dimensions increases.

5 Conclusion

In this paper, we proposed a novel system for alert correlation to identify the multi-stage attacks. The central idea of the proposed system focuses on the introduction of a closed multi-dimensional sequential patterns mining algorithm. The CMD_PrefixSpan algorithm mines sequential patterns from the alert sequences to form multi-stage behavior patterns. Carried out experiments showed the effectiveness of the introduced approach.

Within a dynamic network environment, the novel multi-stage attacks appear continuously. Thus, a future work will include the use of incremental mining algorithm to detect recently appeared attacks.

References

1. Brahmi, H., Brahmi, I., Ben Yahia, S.: Nouvelle Approche de Corrélation d'Alertes basée sur la Fouille Multidimensionnelle. In: Actes des 8èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA), Bordeaux, France, pp. 93–102 (2012)
2. Cuppens, F., Miège, A.: Alert Correlation in a Cooperative Intrusion Detection Framework. In: Proceedings of the IEEE Symposium on Security and Privacy, Berkeley, California, USA, pp. 202–215 (2002)
3. Huang, G., Zuo, N., Ren, J.: Mining Web Frequent Multi-dimensional Sequential Patterns. Information Technology Journal 10(12), 2434 (2011)
4. Li, W., Zhi-tang, L., Jun, F.: Learning Attack Strategies Through Attack Sequence Mining Method. In: Proceedings of the International Conference on Communication Technology (ICCT), Guilin, China, pp. 1–4 (2006)
5. Ning, P., Cui, Y., Reeves, D.S., Xu, D.: Techniques and Tools for Analyzing Intrusion Alerts. Journal ACM Transactions on Information and System Security 7(2), 274–318 (2004)
6. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Journal of Information Systems 24(1), 25–46 (1999)
7. Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.-C.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In: Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany, pp. 215–224 (2001)
8. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and performance Improvements. In: Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT), Avignon, France, pp. 3–17 (1996)
9. Vijayalakshmi, S., Mohan, V., Raja, S.S.: Mining Constraint-based Multidimensional Frequent Sequential Pattern in Web Logs. European Journal of Scientific Research 36(3), 480–490 (2009)

CiDHouse: Contextual SemantIc Data WareHouses

Selma Khouri[1,3], Lama El Saraj[2,4], Ladjel Bellatreche[3], Bernard Espinasse[2], Nabila Berkani[1], Sophie Rodier[4], and Thérèse Libourel[5]

[1] ESI, Algiers, Algeria
{s_khouri,n_berkani}@esi.dz
[2] LSIS, Marseille, France
{lamaelsarraj,bernardespinasse}@lsis.org
[3] LIAS/ISAE-ENSMA, Futuroscope, France
{selma.khouri,bellatreche}@ensma.fr
[4] Assistance publique des Hôpitaux de Marseille, France
{lama.elsaraj,sophie.rodier}@ap-hm.fr
[5] LIRMM, Montpellier, France
therese.libourel@lirmm.fr

Abstract. Dealing with contextualized data is a key challenge in data warehouses ($\mathcal{DW}$). Nowadays, $\mathcal{DW}$ systems are often monocontext. However, in real life applications, $\mathcal{DW}$ indicators are shared by many users with different profiles. In this paper, we propose an ontology-based approach for designing multi-contextual $\mathcal{DW}$. An ontology formalism incorporating the contextualization concepts is given. We propose to consider the contextualization at the conceptual level. We validate our proposal using a real case study from the medical domain.

1 Introduction

Nowadays, the $\mathcal{DW}$ technology becomes an incontestable technology and tool for businesses and organizations to make strategic decisions. One of the difficulties of building a $\mathcal{DW}$ application is handling the heterogeneity of sources. Ontologies play an important role to reduce this heterogeneity and to reason on the ontological concepts. Note that ontologies have been widely developed in several domains such as medicine, engineering, etc. This development motivates the $\mathcal{DW}$ community to consider ontologies in the design steps of the $\mathcal{DW}$, especially in the conceptual modeling and ETL phases [8,1]. During the conceptual phase the ontologies may represent the global schema of the $\mathcal{DW}$ [10]. Some other works proposed then to attach an ontology (usually called local ontology) to each source participating in the $\mathcal{DW}$ construction, and to define mappings between the global and local ontologies [10]. The $\mathcal{DW}$ considering sources embedding ontologies in their repository (usually called semantic databases (SDB)) correspond to this architecture.

In the first era of the $\mathcal{DW}$, the business applications were mono-contextual. A $\mathcal{DW}$ is exploited by multi-context users. To illustrate this proposal, we consider a case study related to the *public hospitals of Marseille, France*. Their

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 458–465, 2013.

services contain information about sojourn duration, coding exhaustiveness rate for each patient per Diagnosis Related Groups, per time dimension, per institution structure, per weight, per age and per international classification of diseases. This domain is formally defined by an ontology that we constructed and validated with the hospital staff. To give visibility about the context we present the following example concerning the sojourn duration per service and per month. In real life, a department chief could find that measures values are incoherent or insufficient to the reality, because sojourn duration is calculated differently from a context to another. A patient may make several sojourns of different durations in different services. Context is represented here by the service where the patient was hospitalized. Consequently, users' needs and interpretation depend on the context.

There is a paradox in the organizations, they need to construct a $\mathcal{DW}$ to store data from heterogeneous sources but the context heterogeneity of end users is not taken into consideration during the construction time. The $\mathcal{DW}$ end-users have a rich knowledge warehouse regarding the context, measure unity, etc. that should be formalized and integrated in the ontologies. A couple of research efforts have proposed contextual ontologies [6,11]. In this work, we present a novel approach based on ontologies to propose a Contextual-SemantIc Data WareHouses (CiDHouse) design. We tested our approach in the healthcare domain in collaboration with the Marseille hospitals. To the best of our knowledge, we are the first to propose an ontology-based approach for multi-context $\mathcal{DW}$ covering the conceptual, logical and ETL phases.

The paper is organized as follows. Section 2 presents the related work. Section 3 presents our proposal. Section 4 concludes the paper.

2 Related Work

Historically, the notion of context was used to indicate 'the part of speech around the content that can explain its significance'. According to the literature, we classify the state of the art studies in two categories: naive approaches and algorithmic approaches. Naive approaches are based on logic properties annotated by the context. Algorithmic approaches propose algorithms to adapt the results of a system to a context. The proposed approaches give solutions for the two phases of the $\mathcal{DW}$: the conceptual and the exploitation level. In order to compare the previous approaches, we identified the following criteria: (1) *C1 : Multi-context exploitation techniques*: this criterion identifies if the approach proposes techniques to facilitate multi-context $\mathcal{DW}$ exploitation (at the conceptual level or at the physical level). (2) *C2: Multi-context measures affected by external entities to the $\mathcal{DW}$*. (3) *C3: Multi-context measures influenced by entities internal to the $\mathcal{DW}$* (Eg. contextual hierarchies). (4) *C4: Knowledge about the contents.* (5) *C5: Semantic variation*: the interpretation, the identification, the expression of measures and dimensions could differ from a user to another, depending on users-context of analysis.

The following table lists the previous approaches in order by approach category, the $\mathcal{DW}$ phases, approach reference, approach description, and criterion

(C1,C2,C3,C4, C5). For our approach, we extended the ontology-based conceptual method for $\mathcal{DW}$ presented in [1] to assist users to model multi-context $\mathcal{DW}$.

Table 1. Related work comparison

$\mathcal{DW}$ phase	Reference	Description	Category	C1	C2	C3	C4	C5
Conceptual level	CiDHouse	Ontology-based for multi-context $\mathcal{DW}$	Ontological	x	x	x	x	x
Conceptual level	[7]	Flexibility and expressivity of hierarchies	Naive	x		x		
$\mathcal{DW}$ exploitation	[4]	Visualization of personalized multidimensional model	Naive	x	x			
$\mathcal{DW}$ exploitation	[5]	Contextualization with document	Naive	x		x	x	
$\mathcal{DW}$ exploitation	[9]	Algorithm to adapt system to the context	Algorithmic	x	x			

3 Our Proposal

Our $\mathcal{DW}$ design follows the hybrid approach, where data sources and user requirements are considered as inputs of the method. Our proposal is based on four foundations that we discuss in the next sections:

3.1 Formalisation of Ontologies Integrating Context

Different languages have been defined to describe ontologies. OWL language is the language recommended by W3C consortium for defining ontologies. Description Logics (DLs) present the formalism underlying OWL language. We thus use DLs as the basic formalism for specifying the framework. In DL, structured knowledge is described using *concepts* denoting unary predicates and *roles* denoting binary predicates. Concepts denote sets of individuals, and roles denote binary relationships between individuals. Concepts can be defined based on other concepts by applying suitable DL constructors (eg. intersection, value restriction, limited existential quantification, etc), equipped with a precise set-theoretic semantics. Based on these definitions, an ontology O is formally defined as follows: O :<*C, R, Ref (C), formalism*>, such that: C: denotes *Concepts* of the model (atomic concepts and concept descriptions); R denotes *Roles* of the model. Roles can be relationships relating concepts to other concepts, or relationships relating concepts to data-values; Ref: $C \rightarrow$ (*Operator*, *Exp(C,R)*). Ref is a *function* defining terminological axioms of a DL TBox. *Operators* can be inclusion ($\sqsubseteq$) or equality ($\equiv$). *Exp(C,R)* is an expression over concepts and roles of O using constructors of DLs such as union, intersection, restriction, etc. and $Formalism$: is the *formalism* followed by the global ontology model like OWL-DL, OWL-LITE, DL-Lite, etc.

We extend this formalization for handling contextual information. Contextual information is described in literature as dependence relationships between

contextualizing properties and contextualized properties. For example, the *Hospitalization duration* property is dependent on the *Medical Unit* and *IdPatient* properties. *Medical Unit* and *IdPatient* are contextualizing properties and *Hospitalization duration* is a contextualized property. We extended the proposed formalization by considering the contextualizing and contextualized properties, as follows: $O :< C, R, Ref(C)$, **Ref**$_{Context}$**, Mes (R)**, $formalism >$ such as:
Ref$_{Context}$ is defined as a mapping relation from the power set of R onto R : $C \times 2^R \rightarrow C \times R$. The properties defined in the left side of the mapping represent *contextualizing* properties, and the properties defined in the right side of the mapping is the *contextualized* property. For the example given above, we would have: $Ref_{Context} : (MedicalUnit, MedicalUnitId), (Patient, IdPatient) \rightarrow (Time, Hospitalizationduration)$.
Mes (R) denotes a function $R \rightarrow Unit$ to define the unit of measurement of each role R.

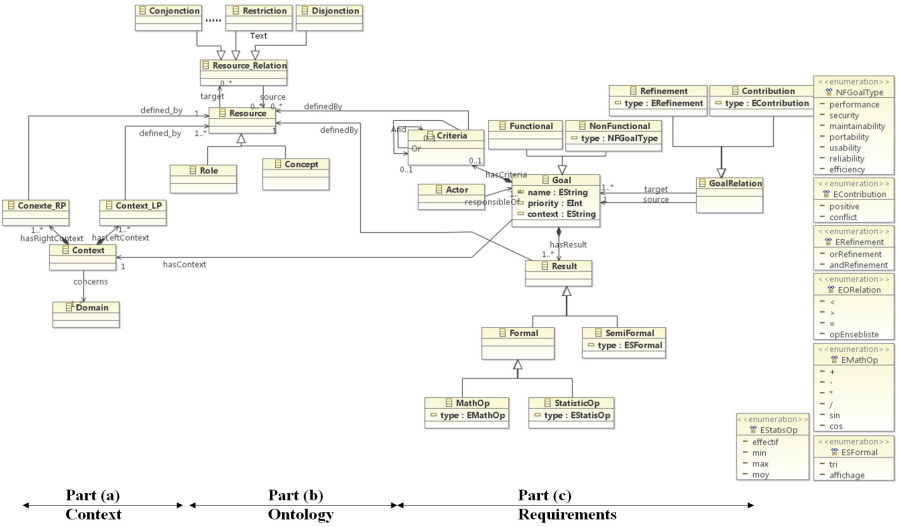


Fig. 1. Ontology model integrating context linked to the context model proposed

Figure 1 illustrates the extension of the ontology model with the contextual model (part (a) and (b)).

3.2 User Requirements Defined on Multi-contextual Ontologies

User requirements are expressed on the ontological level by the means of the goal oriented paradigm. Goal driven analysis has been frequently used for $\mathcal{DW}$ development. It identifies goals and objectives that guide decisions of the organization at different levels. A goal is an objective that the system under consideration should achieve. User's goals have a significant role in our method. They are used

to identify the most relevant data to materialize in the $\mathcal{DW}$. We proposed a goal model considered as a pivot model since it combines three widespread goal-oriented approaches: *KAOS*, *Tropos* and *iStar*. The model is presented in Fig. 1 (Part (C)).

3.3 Definition of the Integration Framework for Integrating Sources Containing Contextual Data

We define an integration framework $<\mathcal{G}, \mathcal{S}, \mathcal{M}>$ adapted to contextual data.

The Global Schema *G*: Schema $\mathcal{G}$ is represented by a Global Ontology (GO). As explained, the global ontology is formally defined to handle contextual information as follows $O :< C, R, Ref(C), Ref_{Context}, Mes(R), formalism >$. Note that the definition of the *GO* concerns only its TBox, which is usually assumed in data integration systems.

The Sources *S*: The set of sources considered are semantic databases ($\mathcal{SDB}$s). Each $\mathcal{SDB}$ is defined by its local ontology (O_i) and its instances part (the ABOX). As, explained previously, the ontology model and its instances can be stored using different storage layouts. $\mathcal{SDB}$s may have different architectures. A $\mathcal{SDB}$ is formally defined as follows $< O_i, I, Pop_c, SL_{O_i}, SL_I, Ar >$ where:

- O_i: $< C, R, Ref, Ref_{context}, Mes(R), formalism >$ is the *ontology* model of the $\mathcal{SDB}$.
- I: presents the *instances* (the ABox) of the $\mathcal{SDB}$.
- $Pop_C : C \rightarrow 2^I$ is a *function* that relates each concept to its instances.
- SL_{O_i}: is the *Storage Layout* of the ontology model. We distinguish three main *relational* representations: *vertical*, *binary* and *horizontal*. Vertical representation stores data in a unique table of three columns (subject, predicate, object). In a binary representation, classes and properties are stored in different tables. Horizontal representation translates each class to table having a column for each property of the class.
- SL_I: is the *Storage Layout* of the instances I. The storage layout used for instances can be the same layout used for storing the ontology, or a different one.
- Ar: is the *architecture* of the $\mathcal{SDB}$.

The Mappings *M*: The mapping assertions are formally defined as follows M:< MapSchemaG, MapSchemaS, MapElmG, MapElmS, Interpretation, SemanticRelation >. This formalization is based on the mapping meta-model presented in [2]:

- MapSchemaG and MapSchemaS: present respectively the *mappable schema* of the global and the local ontology.
- MapElmG and MapElmS: present respectively a *mappable element* of the global and the local ontology schema. This element can be a simple concept, instance or an expression (*Exp*) over the schema.

– Interpretation: presents the *Intentional* interpretation or *Extensional* interpretation of the mapping. In our study, the availability of global and local ontologies allows to define intentional mappings.
– SemanticRelation: three relationships are possible: *Equivalence*, *Containment* or *Overlap*.

3.4 Design Method

We propose a method for designing a semantic $\mathcal{DW}$ covering the following steps: requirements analysis, conceptual design, logical design, ETL process, deployment and physical design. Fig. 2 illustrates these steps.

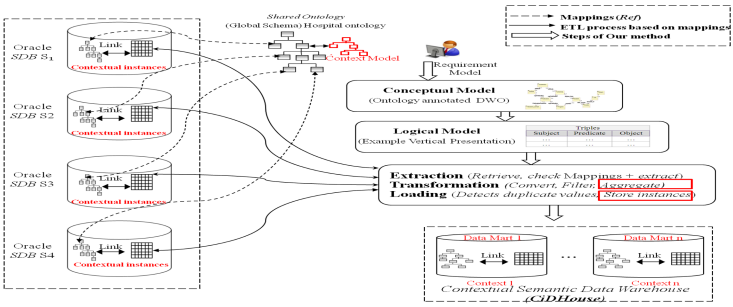


Fig. 2. Design method proposed

Requirements Analysis: This step allows the designer identifying the following: the set of relevant properties used by the target application and the set of treatments it should answer. The first set allows the construction of the *dictionary* containing the relevant concepts required for the application. We defined a connection between coordinates of each goal (*Result* and *Criteria*) and the resources (*concepts* and *roles*) of the *GO* (Fig. 1- Part B and C). This allows the designer to choose the most relevant ontological concepts to express user's goals. Knowing that the *GO* is linked to the data sources, these concepts chosen to express goals inform the designer about the most relevant data to store in the $\mathcal{DW}$ model.

Conceptual Design: A $\mathcal{DW}$ ontology (*DWO*) (that can be viewed as a conceptual abstraction of the $\mathcal{DW}$) is defined from the global ontology (GO) by extracting all concepts and properties used by user goals. Three scenarios materialize this definition: (1) $DWO = GO$: the *GO* corresponds exactly to users' requirements, (2) $DWO \subset GO$: the *DWO* is extracted from the *GO*, (3) $DWO \supset GO$: the *GO* does not fulfill all users' requirements.

Logical Design: The logical $\mathcal{DW}$ model is generated by translating the *DWO* to a relational model. Several works in the literature proposed methods for translating ontologies described in a given formalism (PLIB, OWL, RDF) to a relational or an object-relational representations [3].

ETL Process: The goal of the ETL process is to populate the target $\mathcal{DW}$ schema obtained in the previous step, by data from the sources. Skoutas et al. [8] defined ten generic operators typically encountered in an ETL process, which are: Extract (S,C), Retrieve(S,C), Merge(S,I), Union (C,C'), Join (C, C'), Store(S,C, I), DD(I), Filter(S,C,C'), Convert(C,C') and Aggregate(F, C, C').

These operators have to be leveraged to deal with the semantic aspects of sources. Therefore, we proposed an algorithm formalizing the ETL steps (presented in a previous paper [1]) for populating the *DWO* schema. The algorithm is based on the generic conceptual ETL operators presented. They can then be instantiated according to one of the storage layouts: vertical, binary, horizontal. Each operator will correspond to a defined query. SPARQL is the standard query language used for querying ontologies. We translated each operator to a SPARQL query. Some operators are translated using SPARUL (or SPARQL/Update) language.

Select (Aggregate_Function(?Instance) AS ?Aggregate_Function)
Where {?Instance rdf:type namespace:Class. namespace:DatatypeProperty rdfs:domain namespace:Class.
namespace:DatatypeProperty rdfs:label ContextID} *Group By ?Instance*
SELECT ?Instances ?ContextID
Where {?Instances rdf:type namespace:Class. ?DatatypeProperty rdfs:domain namespace: Class.
?DatatypeProperty rdfs:label ?ContextID. *?DatatypeProperty rdfs:label ContextID }*
Insert into staging_table Values (id, SDO_RDF_TRIPLE_S (?Instances, rdf:type , namespace:Class));

The target $\mathcal{DW}$ model defined by our proposal is populated according to a given DBMS. We validated our proposal using Oracle DBMS. An Oracle $\mathcal{SDB}$ is used to store the target CiDHouse model and the ontology defining its semantics. Oracle has incorporated support for languages RDF and OWL in its system to enable its customers to benefit from a management platform for semantic data. Oracle has defined two subclasses of DLs: *OWLSIF* and a richer fragment *OWLPrime*. We used *OWLPrime* fragment. We first instantiated the integration framework $<\mathcal{G}, \mathcal{S}, \mathcal{M}>$ using the hospital ontology and Oracle semantic sources containing contextual instances. Each source is constructed from the hospital ontology, by using a set of equivalence mappings. We then applied the ETL algorithm proposed, based on the ETL operator adapted to deal with the contextual instances. The result of the integration process is a $\mathcal{DW}$ whose schema corresponds to the classes and properties of the hospital ontology, populated by instances selected from Oracle $\mathcal{SDB}$s.

4 Conclusion

Real life applications are usually shared by many users from heterogeneous domains. Consequently, data managed by these applications contain contextual information. We proposed in this paper a design method for $\mathcal{DW}$ applications, that

deals with contextual information from the first phase of $\mathcal{DW}$ design. A formal definition of domain ontologies enriched with context information is proposed. The design method follows five design steps: requirements definition, conceptual design, logical design and ETL process. The method is validated using a practical case study from the medical domain. An interesting issue that has to be considered is the generation of data marts based on user contexts.

References

1. Bellatreche, L., Khouri, S., Berkani, N.: Semantic data warehouse design: From ETL to deployment à la carte. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013, Part II. LNCS, vol. 7826, pp. 64–83. Springer, Heidelberg (2013)
2. Brockmans, S., Haase, P., Serafini, L., Stuckenschmidt, H.: Formal and conceptual comparison of ontology mapping languages. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 267–291. Springer, Heidelberg (2009)
3. Gali, A., Chen, C., Claypool, K., Uceda-Sosa, R.: From ontology to relational databases. In: ER Workshops, pp. 278–289 (2004)
4. Garrigós, I., Pardillo, J., Mazón, J.-N., Trujillo, J.: A conceptual modeling approach for OLAP personalization. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 401–414. Springer, Heidelberg (2009)
5. Pérez, J.M., Berlanga, R., Aramburu, M.J., Pedersen, T.B.: A relevance-extended multi-dimensional model for a data warehouse contextualized with documents. In: DOLAP 2005 (2005)
6. Pierra, G.: Context representation in domain ontologies and its use for semantic integration of data. Journal of Data Semantics (JoDS) 10, 174–211 (2008)
7. Pitarch, Y., Favre, C., Laurent, A., Poncelet, P.: Enhancing flexibility and expressivity of contextual hierarchies. In: Fuzzy Systems, pp. 1–8 (2012)
8. Skoutas, D., Simitsis, A.: Ontology-based conceptual design of ETL processes for both structured and semi-structured data. IJSWIS 3(4), 1–24 (2007)
9. Stefanidis, K., Shabib, N., Nørvåg, K., Krogstie, J.: Contextual recommendations for groups. In: Advances in Conceptual Modeling, pp. 89–97 (2012)
10. Wache, H., Vogele, T., Visser, U., Stuckenschmidtet, H., et al.: Ontology-based integration of information - a survey of existing approaches. In: OIS, pp. 108–117 (2001)
11. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: PERCOMW, pp. 18–22 (2004)

Analysis of Clinical Documents to Enable Semantic Interoperability

Barbara Franz, Andreas Schuler, and Emmanuel Helm

University of Applied Sciences Upper Austria, Hagenberg, Austria
{barbara.franz,andreas.schuler,emmanuel.helm}@fh-hagenberg.at

Abstract. While there is a wealth of information available for each patient in an Electronic Health Record (EHR), information is not optimally organized for efficient use in patient care and there is still a large gap to achieve semantic and process interoperability. Current researches focus either on fully structured or non-structured clinical documents. A pre-analysis of 1000 real-world clinical documents showed, that most clinical documents are provided in a semi-structured way as level 2 HL7 Clinical Document Architecture (CDA) documents. Thus, an analysis framework is presented which also uses this semi-structure in combination with underlying models and metadata provided by the exchanging infrastructures, allowing summarization, comparison and integration of clinical information. The approach was tested using an IHE compliant EHR system. The analysis framework offers good response times and high accuracy levels. However, there is room for improvement considering processing of non-structured text and non-CDA documents.

Keywords: Semantic interoperability, Electronic Health Record, Clinical Document Architecture.

1 Introduction

It is often asserted that adoption and proper implementation of healthcare systems for information exchange across care settings and integration in electronic health records (EHR) can lead to greater efficiency, better access to quality healthcare, and improve overall health care delivery [1-2]. Although there is a wealth of information available for each patient in an EHR, information is not optimally organized for efficient use in patient care [3-4].

Initiatives like Integrating the Healthcare Enterprise (IHE) or Continua Health Alliance (CHA) define guidelines and propose the use of certain standards like HL7 or DICOM and terminologies like LOINC to facilitate interoperability in healthcare [5-7]. These approaches mainly provide standalone semantic models to cover technical interoperability. A holistic model to allow semantic interoperability and to enable unambiguous exchange of information is still missing [8].

An analysis of 1000 real-world clinical documents showed that most documents are provided in a semi-structured format. Thus, an approach is presented which also uses this semi-structure as well as information provided by (partially) underlying

H. Decker et al. (Eds.): DEXA 2013, Part II, LNCS 8056, pp. 466–473, 2013.

models and metadata provided by the exchanging infrastructures. The framework was designed to analyze clinical documents in EHR systems. It allows the mapping of document contents to an EHR independent model and transformation into document models to enable summarization, comparison and integration of clinical information.

2 Situation Analysis of Clinical Documents in EHR

Standardized international EHR systems are mainly based on IHE. For exchanging healthcare documents, HL7 v3 Clinical Document Architecture Release 2 (CDA R2) is a widely used standard, which also makes documents machine- and human-readable [9]. In few European states, additional approaches like archetype-based OpenEHR are used. Since this approach is also compatible with IHE and HL7 CDA, the focus of this paper is mainly on IHE and CDA.

2.1 Structure of Clinical Documents and Used Vocabulary

Healthcare services use and provide various types of clinical documents – from ePrescription, Care Record Summary (CRS) [10], Continuity of Care Document (CCD), Personal Health Monitoring Record (PHMR), just to name a few – each application domain may produce different outcome. The document content varies depending on the domain, defined policies, the EHR, used guidelines, templates and models. Models comprise [11] available ontologies, terminologies, coding dictionaries and meta thesauri, like for example: Systematized Nomenclature of Medicine - Clinical Terms (SNOMED CT) terminology Description Logic [12], BS/ISO/EN13606 Electronic Health Record for Communications object model [13] and others [14-18].

Independent of the document type, CDA R2 is a worldwide preferred structure for clinical documents [19]. Documents compliant to HL7 v3 CDA R2 are structured into two parts – header and body. The header contains metadata, including assigned codes for classification, combined with information about the author, patient, participants and care context. The actual clinical information is located in the body section of the document. The body can be structured in different complexity levels – from non-structured body (level 1) to a fully structured body sections (level 3) [20]. Level 1 contains no mark-up at all or body sections without codes, level 2 contains sections (identified by code and titles) including mark-up-enriched text, level 3 additionally contains machine readable entry elements, which allow automatic mapping to underlying medical models.

2.2 Pre-analysis of Real-World Clinical Documents

For a situation analysis, 1000 clinical documents were collected over five days in three different Cross Enterprise Document Sharing (XDS) affinity domains at random times. One affinity domain comprises the information systems of two hospitals and 40 physicians. The second affinity domain consists of two hospitals, five mobile nursing care providers and ten nursing homes. The third affinity domain is mainly used for the

exchange of personal health monitoring reports between patients, a hospital, an ambulant cardiologic rehabilitation center, one mobile nursing service and three residential houses with assisted living. The yielded results are outlined in table 1. The analysis of the structure and content of the collected documents showed that most documents are valid CDA documents, but use of other underlying models is only marginal. LOINC Codes are provided in nearly all documents (at least as document type), but especially SNOMED-CT or UMLS are not used. This confirms that approaches solely based on fully structured or non-structured information are not applicable for current EHR systems.

Table 1. Analysis of clinical documents in three affinity domains

Document type	N	Structure
Medical discharge summary (34106-5)	277	Mostly semi-structured data: tables including medical diagnosis, partially in ICD-10, and procedures, intermixed with free text sections containing listings of administered drugs
Laboratory report (11502-2)	269	Highly structured data: tables, CDA Level 2 and 3 annotations
Medical imaging report (18748-4)	193	Semi-structured data, partially level 3: modality and date/time annotated in a constant pattern, use of abbreviations
Nursing care summary (34745-0)	113	Semi-structured data: tables with free-text nursing diagnosis, patient resources, vital signs , procedures intermixed, with free text sections
Personal health monitoring report	73	Highly structured data: tables and CDA level 3 annotations
Other documents	69	Highly unstructured data: abbreviation rich narratives

3 Analysis Framework

For automatic analysis and integration of semi-structured clinical documents in EHR systems, a framework was implemented, which also uses this semi-structure as well as information provided by (partially) underlying models and metadata provided by the exchanging infrastructures. In the following, a brief overview of the framework, its components and functionality is given (see Fig. 1).

The clinical documents are retrieved from IHE compliant EHR in a standard-compliant way. Using metadata provided by the IHE infrastructure, especially by the XDS registry, EHR independent model (as already described in [21]) and a rule engine are initialized. After pre-processing the document, the extracted content is mapped to the derived models, e.g. SNOMED CT, BS/ISO/EN13606 or others [11-18], which allows for transformation into other document models and enables summarization, comparison and integration of clinical information. This can be used to give

healthcare professionals an adequate overview of certain healthcare information, for example over vital signs in comparison with current medication, which can be provided to the user via a web-based frontend.

3.1 Initialization Component for Model and Rule Engine

Using XDS metadata, the EHR independent model can be initialized before the information retrieval process. The type of document can be determined using this metadata, which also allows initialization of the rule engine [22].

The rules are expressed in RuleML [23], which allows the definition of additional rules by a developer in consultation with a domain specialist. Since the effort for the generation of rules, which are needed for the automatic document extraction, for example to provide overviews over vital signs in combination with medication, grows potentially with additional document types and information, the rules are generated semi-automatically using the RIM, the document metadata, HL7 templates, predefined Schematron files and CDA implementation guidelines which can be applied to the various document types.

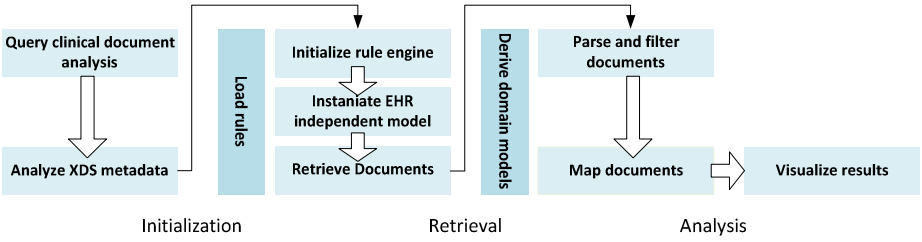


Fig. 1. Overview over analysis

3.2 Information Retrieval Component

Conformance to IHE integration profiles and adherence to suggested standards are preconditions for smooth integration. For information retrieval, the Open Health Tools (OHT) framework [24] is used, which provides free available implementations of several IHE profiles like Patient Identifier Cross-Referencing (PIX), Patient Demographic Query (PDQ), and XDS.

Depending on the requested information, all documents for a patient or only certain document types are retrieved and provided for the information extraction process. For example documents can be retrieved using a PIX identifier, which represents the patient, as well as specified document metadata. For example, in addition to the regular XDS and PIX parameters, the user provides rules requests the generation of a summary for a defined period of the patient's medical history. The retrieval component accesses all documents of the specified patient, between the defined lower and upper creation date, which can be found in the XDS metadata.

3.3 Analysis Component

Due to the absence of an up-to-date medical terminology like SNOMED-CT or the UMLS with semantic qualities [25], a sole knowledge-based approach like Semantic Representation of Medical Documents (SeReMeD) [26] is not feasible. A purely probabilistic approach requires Part of Speech (POS) tagging in order to work properly. Since free text narrative of medical documentation is written in a rather dense abbreviation-rich style, as confirmed by the initial document analysis, this approach was dismissed considering the efforts required for creating a gold standard set of documents for training purposes [27].

State of the art semantic and syntactic information extraction approaches, for example described in [28] require an extensive high quality knowledge base and do not take advantage of the at least partially structured nature of CDA documents. Thus, a hybrid model- and rule-based approach as described below, leveraging structured as well as semi-structured content is used.

Initially after retrieval, documents are clustered according to their document type, and further parameters like used code systems, codes or title (depending on the applied set of rules). A list of documents embedded in a map leveraging the document type code or configured name as a key is returned. All retrieved clinical documents are parsed according to the predefined rules based on the document type. This preprocessing step includes the application of tokenization and stemming algorithms. Afterwards, available processing rules, which define a set of implicit or explicit scopes, are executed. For example, to retrieve relevant information, each document is inspected for matching code, title or templateId properties. In case of an exact code or fuzzy title match a document is considered as relevant. In a second step, the matching process is repeated for the document's sections. On completion of the filtering process, relevant documents are assembled in a list for further processing.

The relevant documents are weighted considering containing terms using an inverted index generated using normalized TF-IDF weights. Depending on the user's request, a lookup in the index is conducted and extracted information is set into context using the underlying models.

4 Evaluation

The concept was tested using an existing IHE compliant system based on a use case in the field of cardiologic rehabilitation. Within the scope of the use case a patient has to monitor his/her vital signs at home using a CHA compliant telemonitoring solution. Medical personnel in the ambulant cardiologic rehabilitation center can then access the vital signs stored as a PHMR. During the rehab process it is important for the caretakers to not only know the patient's vital signs over a certain time period, but also whether the vital signs are influenced by administered medication.

Each application domain may produce different outcome documents, which can all be stored in the repository using XDS. This example focuses on the retrieval of three different document types, identified by the metadata typeId, stored in the format of HL7 v3 CDA R2. Due to missing semantic interoperability, without the proposed

solution, the medical personnel would have to retrieve each document in the EHR and manually search for all documented vital signs as well as all medication which are currently administered and manually compare them.

4.1 Proof of Concept

Thus, in collaboration with medical professionals from 13 institutions, hitherto existing documentation and processes were analyzed. Rules for documentation and communication were developed and applied to the IHE compliant EHR system [29-30]. This environment allows the exchange of HL7 CDA documents between healthcare providers in the region. The environment was extended in order to support the integration of the CHA compliant telemonitoring solution. Hence measured vital data are transformed into a standardized PHMR and stored in an IHE repository.

When the medical personnel requests vital signs in context of administered medication, the rule engine and EHR independent model are initialized as described above. A model for PHMR and one for discharge summaries are derived. The retrieval component fetches all PHMR and discharge summaries, which are then pre-processed by the analysis component. All vital signs data are analysed according to the pre-defined rules. Thus vital signs found in sections with LOINC-code 8716-3 in PHMR are mapped to PHMR model, medication and vital signs in discharge summaries to the applicable model etc. Vital signs for example are clustered by the identified type of vital signs and transformations are performed in respect to the defined rules.

4.2 Evaluation Results

During the first run a significant spike in response times could be observed for all tests. Performance throughout the remaining runs was consistent between 2500 ms to 3500 ms (runs with retrieval component) and 3000 ms to 3500 ms (analysis component). From ten discharge summaries, one document didn't contain any of the processable information types. Thus, only nine were used for the generation of the vital signs summary.

The analysis component recognized 68 from 69 medications in the discharge summary, 15 from 15 vital signs, but only 117 from 122 diagnoses, since diagnoses written in multiple rows were interpreted as several diagnoses. Due to the fully structured PHMR (level 3) all vital signs were recognized.

5 Discussion and Conclusion

The runtime performance of the document retrieval process is highly time-consuming. This is caused by the OHT service used for XDS query and retrieval as well as XCA. The parallelisation of the XDS communication might reduce runtime up to 50%. A better solution would be prefetching (preloading) of documents. This is technically possible, but this will lead to legal concerns due to authentication, logging and missing care context.

The presented results are quite promising. Nevertheless, a subtle variation in accuracy can be observed. This variation can be related to the degree of structure of the underlying documents as already stated in table 1. An improvement in abbreviation expansion, spell error correction and localization, should also improve the accuracy of analysis of non-structured text.

The approach presented in this paper shows an application that provides ready access to a collection of information useful in healthcare. The system is designed for efficient use during patient care services. The data retrieval and analysis components enable a summarization of several healthcare documents of a patient. Thus, healthcare professionals don't need to search several documents, but can easily get an overview over relevant healthcare data.

The developed system presented in this paper implements a modular rule-based information extraction algorithm leveraging semi-structured contents like tables and listings of CDA documents. This approach shows considerable performance advantages when applied to rather highly structured as well as semi-structured data but has some drawbacks when structuredness decreases, thus offering room for improvement considering processing of non-structured text and non-CDA formats.

Acknowledgement. This research has been partially funded by the Austrian Research Promotion Agency (FFG), the European Regional Development Fund (ERDF) in cooperation with the Upper Austrian state government (REGIO 13, Innovative Upper Austria, Health Cluster).

References

1. Chaudhry, B., et al.: Systematic Review: Impact of Health Information Technology on Quality, Efficiency, and Costs of Medical Care. Ann. Intern. Med. 144(10), 742–752 (2006)
2. Thakkar, M., Davis, D.: Risks, Barriers, and Benefits of EHR Systems: A Comparative Study Based on Size of Hospital. Perspect Health Inf. Manag. 3, 5 (2006)
3. Green, M.L., Ciampi, M.A., Ellis, P.J.: Residents' medical information needs in clinic: are they being met? Am. J. Med. 109, 218–223 (2000)
4. Barnett, G.O., et al.: Overcoming Information Overload: An Information System for the Primary Care Physician. In: Fieschi, M., Coiera, E., Li, Y.C.J. (eds.) Proceedings from the Medinfo 2004 World Congress on Medical Informatics. Studies in Health Technology and Informatics, vol. 107, pp. 273–276. AMIA, IOS Press (2004)
5. Health Level Seven: HL7. Health Level Seven (2013), `http://www.hl7.org`
6. Heitmann, K., Gobrecht, K-H.: HL7 Kommunikationsstandards für das Gesundheitswesen. Ein Überblick, HL7 Benutzergruppe in Deutschland, LUP, Köln (2009)
7. IHE International: IHE International: Integrating the Healthcare Enterprise (2013), `http://www.ihe.net`
8. Bernal, J., Lopez, D., Blobel, B.: Architectural approach for semantic EHR systems development based on Detailed Clinical Models Stud Health Technol. Inform. (2012)
9. Spronk, R.: CDA. Clinical Document Architecture (2007), `http://hl7book.net`

10. IHE International: IHE Information Technology Infrastructure – Technical Framework, Volume 1 Integration Profiles (2013), http://www.ihe.net/Technical_Framework/index.cfm#IT
11. Blobel, B.G., Engel, K., Pharow, P.: Semantic interoperability–HL7 Version 3 compared to advanced architecture standards. Methods Inf. Med. 45(4), 343–353 (2006)
12. Benson, T.: Principles of Health Interoperability HL7 and SNOMED, HI. Springer, London (2010), doi:10.1007/978-1-84882-803-2_9
13. ISO: EN13606 Health informatics - Electronic health record communication - Part 1: Reference model (ISO 13606-1:2008); English version EN ISO 13606-1:2012 (2012)
14. HL7 International: HL7 Version 3 Standard: Data Types - Abstract Specification, Release 2 (2012), http://www.hl7.org
15. Schadow, G., McDonald, C.J.: The Unified Code for Units of Measure, Regenstrief Institute (2009), http://unitsofmeasure.org/ucum.html
16. NIH: RxNorm (2013), http://www.nlm.nih.gov/research/umls/rxnorm/
17. WHO. International Classification of Diseases (2013), http://www.who.int/classifications/icd/en/
18. NIH: Unified Medical Language System (2013), http://www.nlm.nih.gov/research/umls/
19. Health Level Seven: HL7 Reference Information Model (2012), http://www.hl7.org/implement/standards/rim.cfm
20. Dolin, R., et al.: HL7 Clinical Document Architecture, Release 2. J. Am. Med. Inform. Assoc. 13, 30–39 (2006)
21. Franz, B., Mayr, H.: Providing Semantic Interoperability for Integrated Healthcare Using a Model Transformation Approach. In: Proceedings EMSS 2011, Rome, Italy (2011)
22. Friedmann-Hill, E.: Jess® The Rule Engine for the Java™ Platform, Version 7.1p2, Sandia National Laboratories (November 2008)
23. Boley, H., et. al.: Schema Specification of Deliberation RuleML, Version 1.0 (April 2012)
24. OpenHealthTools, 2011. OpenHealthTools (2012), http://www.openhealthtools.org
25. UMLS Unified Medical Language System, http://www.nlm.nih.gov/research/umls/ (visited on April 06, 2012)
26. Denecke, K.: Semantic structuring of and information extraction from medical documents using the UMLS. Methods Inf. Med. 47(5), 425–434 (2008)
27. Paizoni, T.: A Rule-Based Summarization Approach for CDA Documents in an IHE compliant System, Hagenberg, Austria (2012)
28. Sunita, S.: Foundations and Trends in Databases – Information Extraction, vol. 1. 3. Now Publishers (2008)
29. Franz, B., Lehner, M., Mayr, H., Mayr, H.: e-Care – IHE-Compliant Patient-Data Exchange in Order to Enable Home & Mobile Care of Elderly People Within an e-Care Affinity Domain. In: Proceedings 6th International Conference on Information Technology: New Generations, Las Vegas, April 25-27 (2009)
30. Franz, B., Schuler, A., Buchmayr, M.: Context-enriched personal health monitoring. In: Proceedings AAL Kongress 2013, Berlin, Germany (2013)
31. Stenzhorn, H., Pacheco, E.J., Nohama, P., Schulz, S.: Automatic Mapping of Clinical Documentation to SNOMED CT. In: MIE 2009, pp. 228–232 (2009)

Author Index