Abdelkader Hameurlain
Stephen W. Liddle
Klaus-Dieter Schewe
Xiaofang Zhou (Eds.)

LNCS 6860

Database and Expert Systems Applications

22nd International Conference, DEXA 2011
Toulouse, France, August/September 2011
Proceedings, Part I

Springer

Lecture Notes in Computer Science 6860

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

Abdelkader Hameurlain Stephen W. Liddle
Klaus-Dieter Schewe Xiaofang Zhou (Eds.)

Database and Expert Systems Applications

22nd International Conference, DEXA 2011
Toulouse, France, August 29 - September 2, 2011
Proceedings, Part I

Springer

Volume Editors

Abdelkader Hameurlain
Paul Sabatier University, IRIT Institut de Recherche en Informatique de Toulouse
118, route de Narbonne, 31062 Toulouse Cedex, France
E-mail: hameur@irit.fr

Stephen W. Liddle
Brigham Young University, 784 TNRB
Provo, UT 84602, USA
E-mail: liddle@byu.edu

Klaus-Dieter Schewe
Software Competence Centre Hagenberg
Softwarepark 21, 4232 Hagenberg, Austria
E-mail: kd.schewe@scch.at

Xiaofang Zhou
University of Queensland, School of Information Technology
and Electrical Engineering
Brisbane QLD 4072, Australia
E-mail: zxf@uq.edu.au

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-23087-5 e-ISBN 978-3-642-23088-2
DOI 10.1007/978-3-642-23088-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011934284

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

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

This volume includes invited papers, research papers, and short papers presented at DEXA 2011, the 22^{nd} International Conference on Database and Expert Systems Applications, held in Toulouse, France. DEXA 2011 continued the long and successful DEXA tradition begun in 1990, bringing together a large collection of bright researchers, scientists, and practitioners from around the world to share new results in the areas of database, intelligent systems, and related advanced applications.

The call for papers resulted in the submission of 207 papers, of which 52 were accepted as regular research papers, and 40 were accepted as short papers. The authors of these papers come from 47 different countries. These papers discuss a range of topics including:

- Query processing and Skyline queries
- Search (Web, Semantic Web, database)
- Data integration, transactions, optimization, and design
- Physical aspects of databases, storage
- Database semantics
- Security and privacy
- Spatial and temporal data
- Semantic Web
- Web applications
- Data mining
- Ontologies
- Distribution
- Information retrieval
- XML querying and views
- Business applications
- User support

Three internationally recognized scholars submitted papers and delivered keynote speeches:

Patrick Valduriez: Principles of Distributed Data Management 2020?

Bernhard Thalheim: The Science of Conceptual Modelling

Gabriele Kern-Isberner: Probabilistic Logics in Expert Systems: Approaches, Implementations, and Applications

In addition to the main conference track, DEXA 2011 also included 12 workshops that explored the conference theme within the context of life sciences, specific application areas, and theoretical underpinnings.

We are grateful to the hundreds of authors who submitted papers to DEXA 2011 and to our large Program Committee for the many hours they spent carefully reading and reviewing these papers. The Program Committee was also assisted by a number of external referees, and we appreciate their contributions and detailed comments.

We are thankful to the Institut de Recherche en Informatique de Toulouse at the Université Paul Sabatier for organizing DEXA 2011, and for the excellent working atmosphere provided. In particular, we recognize the efforts of the conference Organizing Committee, including Makoto Takizawa (Seikei University, Japan; Honorary Chairperson), Abdelkader Hameurlain (IRIT, Paul Sabatier University, France; General Chair), Riad Mokadem (IRIT, Paul Sabatier University; Local Organization), Vladimir Marik (Czech Technical University, Czech Republic; Publication Chair), Franck Morvan (IRIT, Paul Sabatier University, Toulouse, France; Workshops Co-chair), A Min Tjoa (Technical University of Vienna, Austria; Workshops Co-chair), and Roland R. Wagner (FAW, University of Linz, Austria; Workshops Co-chair). Without the diligent efforts of these people, DEXA 2011 would not have been possible.

Finally, we are especially grateful to Gabriela Wagner, whose professional attention to detail and skillful handling of all aspects of the Program Committee management and proceedings preparation was most helpful.

August 2011

Stephen W. Liddle
Klaus-Dieter Schewe
Xiaofang Zhou

Program Committee

Honorary Chairperson

Makoto Takizawa	Seikei University, Japan

General Chair

Abdelkader Hameurlain	IRIT, Paul Sabatier University, Toulouse, France

Conference Program Chair

Stephen Liddle	Brigham Young University, USA
Klaus-Dieter Schewe	Software Competence Center Hagenberg and Johannes Kepler University Linz, Austria
Xiaofang Zhou	University of Queensland, Australia

Program Committee

Witold Abramowicz	The Poznan University of Economics, Poland
Osman Abul	TOBB University, Turkey
Rafael Accorsi	University of Freiburg, Germany
Hamideh Afsarmanesh	University of Amsterdam, The Netherlands
Riccardo Albertoni	CNR-IMATI-GE, Italy
Toshiyuki Amagasa	University of Tsukuba, Japan
Rachid Anane	Coventry University, UK
Annalisa Appice	Università degli Studi di Bari, Italy
Mustafa Atay	Winston-Salem State University, USA
James Bailey	University of Melbourne, Australia
Spiridon Bakiras	City University of New York, USA
Ladjel Bellatreche	ENSMA-Poitiers University, France
Morad Benyoucef	University of Ottawa, Canada
Catherine Berrut	Grenoble University, France
Bishwaranjan Bhattacharjee	IBM Thomas J. Watson Research Center, USA
Debmalya Biswas	SAP Research, Germany
Agustinus Borgy Waluyo	Institute for Infocomm Research, Singapore
Patrick Bosc	IRISA/ENSSAT, France
Athman Bouguettaya	CSIRO, Australia
Danielle Boulanger	University of Lyon, France
Omar Boussaid	University of Lyon, France

Stephane Bressan	National University of Singapore, Singapore
Patrick Brezillon	University Paris VI, France
Yingyi Bu	Microsoft, China
Luis M. Camarinha-Matos	Universidade Nova de Lisboa + Uninova, Portugal
Yiwei Cao	RWTH Aachen University, Germany
Barbara Carminati	Università degli Studi dell'Insubria, Italy
Silvana Castano	Università degli Studi di Milano, Italy
Barbara Catania	Università di Genova, Italy
Michelangelo Ceci	University of Bari, Italy
Wojciech Cellary	University of Economics at Poznan, Poland
Cindy Chen	University of Massachusetts Lowell, USA
Phoebe Chen	La Trobe University, Australia
Shu-Ching Chen	Florida International University, USA
Hao Cheng	University of Central Florida, USA
Reynold Cheng	The University of Hong Kong, China
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, China
Oscar Corcho	Universidad Politécnica de Madrid, Spain
Bin Cui	Peking University, China
Emiran Curtmola	University of California, San Diego, USA
Alfredo Cuzzocrea	University of Calabria, Italy
Deborah Dahl	Conversational Technologies, worldwide
Jérôme Darmont	Université Lumière Lyon 2, France
Valeria De Antonellis	Università di Brescia, Italy
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	Universidad Politécnica de 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	Chinese Academy of Sciences, China
Gillian Dobbie	University of Auckland, New Zealand
Peter Dolog	Aalborg University, Denmark
Dejing Dou	University of Oregon, USA
Dirk Draheim	Universität Innsbruck, Austria

Cedric du Mouza	CNAM, France
Johann Eder	University of Vienna, Austria
David Embley	Brigham Young University, USA
Suzanne M. Embury	The University of Manchester, UK
Christian Engelmann	Oak Ridge National Laboratory, USA
Bettina Fazzinga	University of Calabria, Italy
Leonidas Fegaras	The University of Texas at Arlington, USA
Flavio Ferrararotti	Victoria University of Wellington, New Zealand
Stefano Ferilli	University of Bari, Italy
Eduardo Fernandez	Florida Atlantic University, USA
Filomena Ferrucci	Università di Salerno, Italy
Flavius Frasincar	Erasmus University Rotterdam, The Netherlands
Bernhard Freudenthaler	Software Competence Center Hagenberg, Austria
Hiroaki Fukuda	Shibaura Institute of Technology, Japan
Steven Furnell	University of Plymouth, 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
Nathan Griffiths	University of Warwick, UK
Sven Groppe	Lübeck University, Germany
William Grosky	University of Michigan, USA
Volker Gruhn	Leipzig University, Germany
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, Toulouse, France
Ibrahim Hamidah	Universiti Putra Malaysia, Malaysia
Wook-Shin Han	Kyungpook National University, Korea
Takahiro Hara	Osaka University, Japan
Theo Härder	TU Kaiserslautern, Germany
Francisco Herrera	University of Granada, Spain
Steven Hoi	Nanyang Technological University, Singapore
Estevam Rafael Hruschka Jr.	Carnegie Mellon University, USA
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

San-Yih Hwang	National Sun Yat-Sen University, Taiwan
Ionut Emil Iacob	Georgia Southern University, USA
Renato Iannella	Semantic Identity, Australia
Sergio Ilarri	University of Zaragoza, Spain
Abdessamad Imine	University of Nancy, France
Yoshiharu Ishikawa	Nagoya University, Japan
Mizuho Iwaihara	Waseda University, Japan
Adam Jatowt	Kyoto University, Japan
Peiquan Jin	University of Science and Technology, China
Ejub Kajan	State University of Novi Pazar, Serbia
Anne Kao	Boeing Phantom Works, USA
Stefan Katzenbeisser	Technical University of Darmstadt, Germany
Yiping Ke	Chinese University of Hong Kong, Hong Kong
Sang-Wook Kim	Hanyang University, Korea
Markus Kirchberg	Hewlett-Packard Laboratories, Singapore
Hiroyuki Kitagawa	University of Tsukuba, Japan
Carsten Kleiner	University of Applied Sciences and Arts Hannover, Germany
Ibrahim Korpeoglu	Bilkent University, Turkey
Harald Kosch	University of Passau, Germany
Michal Krátký	VSB-Technical University of Ostrava, Czech Republic
Arun Kumar	IBM India Research Lab., India
Ashish Kundu	Purdue University, 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, Université Montpellier 2, France
Mong Li Lee	National University of Singapore, Singapore
Alain Toinon Leger	Orange - France Telecom R&D, France
Daniel Lemire	Université du Québec à Montréal, Canada
Pierre Lévy	Public Health Department, France
Lenka Lhotska	Czech Technical University, Czech Republic
Wenxin Liang	Dalian University of Technology, China
Stephen W. Liddle	Brigham Young University, USA
Lipyeow Lim	IBM T.J. Watson Research Center, USA
Tok Wang Ling	National University of Singapore, Singapore
Sebastian Link	Victoria University of Wellington, New Zealand
Volker Linnemann	University of Lübeck, Germany
Chengfei Liu	Swinburne University of Technology, Australia
Chuan-Ming Liu	National Taipei University of Technology, Taiwan
Fuyu Liu	University of Central Florida, USA

Hong-Cheu Liu	Diwan University, Taiwan
Hua Liu	Xerox Research Center at Webster, USA
Jorge Lloret Gazo	University of Zaragoza, Spain
Miguel Ángel López Carmona	University of Alcalá de Henares, Spain
Peri Loucopoulos	Loughborough University, UK
Chang-Tien Lu	Virginia Tech, USA
Jianguo Lu	University of Windsor, Canada
Alessandra Lumini	University of Bologna, Italy
Cheng Luo	Coppin State University, USA
Hui Ma	Victoria University of Wellington, New Zealand
Qiang Ma	Kyoto University, Japan
Stéphane Maag	TELECOM & Management SudParis, France
Nikos Mamoulis	University of Hong Kong, Hong Kong
Vladimir Marik	Czech Technical University, Czech Republic
Pierre-Francois Marteau	Université de Bretagne Sud, France
Elio Masciari	ICAR-CNR, Italy
Norman May	SAP Research Center, Germany
Jose-Norberto Mazon	University of Alicante, Spain
Dennis McLeod	University of Southern California, USA
Brahim Medjahed	University of Michigan - Dearborn, USA
Harekrishna Misra	Institute of Rural Management Anand, India
Jose Mocito	INESC-ID/FCUL, Portugal
Lars Mönch	FernUniversität in Hagen, Germany
Riad Mokadem	IRIT, Paul Sabatier University, France
Yang-Sae Moon	Kangwon National University, Korea
Reagan Moore	San Diego Supercomputer Center, USA
Franck Morvan	IRIT, Paul Sabatier University, Toulouse, France
Mirco Musolesi	University of Cambridge, UK
Ismael Navas-Delgado	University of Málaga, Spain
Wilfred Ng	University of Science and Technology, Hong Kong
Javier Nieves Acedo	Deusto University, Spain
Selim Nurcan	University Paris 1 Pantheon Sorbonne, France
Mehmet Orgun	Macquarie University, Australia
Mourad Oussalah	University of Nantes, France
Gultekin Ozsoyoglu	Case Western Reserve University, USA
George Pallis	University of Cyprus, Cyprus
Christos Papatheodorou	Ionian University, Corfu, Greece
Marcin Paprzycki	Polish Academy of Sciences, Warsaw Management Academy, Poland
Oscar Pastor	Universidad Politecnica de Valencia, Spain
Jovan Pehcevski	MIT University, Skopje, Macedonia
Reinhard Pichler	Technische Universität Wien, Austria
Clara Pizzuti	CNR, ICAR, Italy

Jaroslav Pokorny	Charles University in Prague, Czech Republic
Giuseppe Polese	University of Salerno, Italy
Pascal Poncelet	LIRMM, France
Elaheh Pourabbas	National Research Council, Italy
Xiaojun Qi	Utah State University, USA
Gerald Quirchmayr	University of Vienna, Austria and University of South Australia, Australia
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	University of Calabria, Italy
Igor Ruiz Agúndez	Deusto University, Spain
Giovanni Maria Sacco	University of Turin, Italy
Shazia Sadiq	University of Queensland, Australia
Simonas Saltenis	Aalborg University, Denmark
Demetrios G Sampson	University of Piraeus, Greece
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
Klaus-Dieter Schewe	Software Competence Centre Hagenberg, Austria
Erich Schweighofer	University of Vienna, Austria
Florence Sedes	IRIT Toulouse, France
Nazha Selmaoui	University of New Caledonia, France
Patrick Siarry	Université Paris 12 (LiSSi), France
Gheorghe Cosmin Silaghi	Babes-Bolyai University of Cluj-Napoca, Romania
Leonid Sokolinsky	South Ural State University, Russia
Bala Srinivasan	Monash University, Australia
Umberto Straccia	Italian National Research Council, Italy
Darijus Strasunskas	Norwegian University of Science and Technology (NTNU), Norway
Lena Stromback	Linköpings Universitet, Sweden
Aixin Sun	Nanyang Technological University, Singapore
Raj Sunderraman	Georgia State University, USA
Ashish Sureka	Infosys Technologies Limited, India
David Taniar	Monash University, Australia
Cui Tao	Brigham Young University, USA

Maguelonne Teisseire	LIRMM, Université Montpellier 2, 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	Ritsumeikan University, Japan
Bernhard Thalheim	Christian-Albrechts-Universität Kiel, Germany
J.M. Thevenin	University of Toulouse, France
Philippe Thiran	University of Namur, Belgium
Helmut Thoma	University of Basel, Switzerland
A. Min Tjoa	Technical University of Vienna, Austria
Vicenc Torra	Universitat Autonoma de Barcelona , Spain
Farouk Toumani	Université Blaise Pascal, France
Traian Truta	Northern Kentucky University, USA
Vassileios Tsetsos	National and Kapodistrian University of Athens, Greece
Theodoros Tzouramanis	University of the Aegean, Greece
Roberto Uribeetxebarria	Mondragon University, Spain
Genoveva Vargas-Solar	LSR-IMAG, France
Maria Vargas-Vera	The Open University, UK
Krishnamurthy Vidyasankar	Memorial University of Newfoundland, Canada
Marco Vieira	University of Coimbra, Portugal
Johanna Völker	University of Mannheim, Germany
Jianyong Wang	Tsinghua University, China
Junhu Wang	Griffith University, Brisbane, Australia
Kewen Wang	Griffith University, Brisbane, Australia
Qing Wang	University of Otago, Dunedin, New Zealand
Wei Wang	University of New South Wales, Sydney, Australia
Wendy Hui Wang	Stevens Institute of Technology, USA
Andreas Wombacher	University of Twente, The Netherlands
Lai Xu	SAP Research, Switzerland
Hsin-Chang Yang	National University of Kaohsiung, Taiwan
Ming Hour Yang	Chung Yuan Christian University, Taiwan
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, Australia
Yanchang Zhao	University of Technology, Sydney, Australia
Yu Zheng	Microsoft Research Asia, China
Xiaofang Zhou	University of Queensland, Australia
Qiang Zhu	The University of Michigan, USA

Yan Zhu	Southwest Jiaotong University, Chengdu, China
Urko Zurutuza	Mondragon University, Spain

External Reviewers

Mohamad Othman Abdallah	University of Grenoble, LIG, France
Sandra de Amo	University of Uberlandia, Brazil
Laurence Rodrigues do Amaral	Federal University of Goias, Brazil
Sandra de Amo	Federal University of Uberlandia, Brazil
Diego Arroyuelo	Yahoo! Research Latin America, Chile
Tigran Avanesov	INRIA Nancy Grand Est, France
Ali Bahrami	Boeing, USA
Zhifeng Bao	National University of Singapore, Singapore
Devis Bianchini	University of Brescia, Italy
Souad Boukhadouma	University of USTHB, Algeria
Paolo Bucciol	UDLAP, LAFMIA, Mexico
Diego Ceccarelli	ISTI-CNR Pisa, Italy
Camelia Constantin	Université Pierre et Marie Curie-Paris VI, Paris, France
Yan Da	University of Science and Technology, Hong Kong
Matthew Damigos	NTUA, Greece
Franca Debole	ISTI-CNR Pisa, Italy
Paul de Vreize	Bournemouth University, UK
Raimundo F. DosSantos	Virginia Tech, USA
Gilles Dubois	MODEME, University of Lyon-Jean Moulin Lyon 3, France
Qiong Fang	University of Science and Technology, Hong Kong
Fabio Fassetti	DEIS, University of Calabria, Italy
Ingo Feinerer	Vienna University of Technology, Austria
Daniel Fleischhacker	University of Mannheim, Germany
Fausto Fleites	Florida International University, USA
Filippo Furfaro	DEIS, University of Calabria, Italy
Claudio Gennaro	ISTI-CNR Pisa, Italy
Panagiotis Gouvas	Ubitech EU
Carmine Gravino	University of Salerno, Italy
Hsin-Yu Ha	Florida International University, USA
Allel Hadj-Ali	IRISA-University of Rennes 1, France
Duong Hoang	Vienna University of Technology, Austria
Enrique de la Hoz	Universidad de Alcala, Spain
Hai Huang	Swinburne University of Technology, Australia
Gilles Hubert	IRIT, Université Paul Sabatier, Université de Toulouse, France
Mohammad Rezwanul Huq	University of Twente, The Netherlands

Saiful Islam	Swinburne University of Technology, Australia
Min-Hee Jang	Hanyang University, Korea
Hélène Jaudoin	IRISA-University of Rennes 1, France
Lili Jiang	Lanzhou University, China
Selma Khouri	ESI, Algeria
Emin Yigit Koksal	Bilkent University, Turkey
Theodorich Kopetzky	SCCH - Software Competence Center Hagenberg, Austria
Olivier le-Goaer	University of Pau, France
Paea LePendu	Stanford University, USA
Kenneth Leung	University of Science and Technology, Hong Kong
Wenxin Liang	Dalian University of Technology, China
Haishan Liu	University of Oregon, USA
Rosanne Liu	The University of Michigan - Dearborn, USA
Xuezheng Liu	Google Inc., USA
Xutong Liu	Virginia Tech, USA
Yannick Loiseau	LIMOS, Blaise Pascal University, France
Carlos Manuel López Enríquez	Grenoble INP - UDLAP, LIG-LAFMIA, France
Min Luo	Tokyo Institute of Technology, Japan
Laura Maag	Alcatel-Lucent Bell-Labs, France
Lucrezia Macchia	University of Bari, Italy
Ivan Marsa-Maestre	Universidad de Alcala, Spain
Michele Melchiori	University of Brescia, Italy
Ruslan Miniakhmetov	South Ural State University, Chelyabinsk, Russia
Ronald Ocampo	Florida International University, USA
Anna Oganyan	Georgia Southern University, USA
Ermelinda Oro	ICAR-CNR, Italy
Francesco Pagliarecci	Università Politecnica delle Marche, Italy
Constantin Pan	South Ural State University, Chelyabinsk, Russia
Vassia Pavlaki	NTUA, Greece
Olivier Pivert	IRISA-University of Rennes 1, France
Xu Pu	Tsinghua University, China
Gang Qian	University of Central Oklahoma, USA
Michele Risi	University of Salerno, Italy
Flavio Rizzolo	Business Intelligence Network (BIN), Canada
Edimilson Batista dos Santos	Federal University of Rio de Janeiro, Brazil
Federica Sarro	University of Salerno, Italy
Vadim Savenkov	Vienna University of Technology, Austria
Wei Shen	Tsinghua University, China
Grégory Smits	IRISA-University of Rennes 1, France
Sofia Stamou	Ionian University, Corfu, Greece

Lubomir Stanchev	Indiana University - Purdue University Fort Wayne, USA
Chad M.S. Steel	Virginia Tech, USA
Thomas Stocker	University of Freiburg, Germany
Nick Amirreza Tahamtan	Vienna University of Technology, Austria
Guilaine Talens	MODEME, University of Lyon-Jean Moulin Lyon 3, France
Lu-An Tang	University of Illinois at Urbana Champaign, USA
Caglar Terzi	Bilkent University, Turkey
Gabriel Tolosa	National University of Lujan, Argentina
Claudio Vairo	ISTI-CNR Pisa, Italy
Changzhou Wang	Boeing, USA
Yousuke Watanabe	Tokyo Institute of Technology, Japan
Andreas Weiner	University of Kaiserslautern, Germany
Yimin Yang	Florida International University, USA
Soek-Ho Yoon	Hanyang University, Korea
Sira Yongchareon	Swinburne University of Technology, Australia
Tomoki Yoshihisa	Osaka University, Japan
Jing Yuan	University of Science and Technology of China, China
Chao Zhu	The University of Michigan - Dearborn, USA
Mikhail Zymbler	South Ural State University, Chelyabinsk, Russia

Table of Contents – Part I

Keynote Talks

Query Processing

Database Semantics

Skyline Queries

Security and Privacy

Spatial and Temporal Data

Semantic Web Search

Storage and Search

Web Search

Data Integration, Transactions and Optimization

Web Applications

Table of Contents – Part II

XML Querying and Views

Data Mining

Queries and Search

Semantic Web

Information Retrieval

Business Applications

User Support

Indexing

Queries, Views and Data Warehouses

Ontologies

Physical Aspects of Databases

Design

Distribution

Miscellaneous Topics

Principles of Distributed Data Management in 2020?*

Patrick Valduriez

INRIA and LIRMM, Montpellier – France
Patrick.Valduriez@inria.fr

Abstract. With the advents of high-speed networks, fast commodity hardware, and the web, distributed data sources have become ubiquitous. The third edition of the Özsu-Valduriez textbook *Principles of Distributed Database Systems* [10] reflects the evolution of distributed data management and distributed database systems. In this new edition, the fundamental principles of distributed data management could be still presented based on the three dimensions of earlier editions: distribution, heterogeneity and autonomy of the data sources. In retrospect, the focus on fundamental principles and generic techniques has been useful not only to understand and teach the material, but also to enable an infinite number of variations. The primary application of these generic techniques has been obviously for distributed and parallel DBMS versions. Today, to support the requirements of important data-intensive applications (e.g. social networks, web data analytics, scientific applications, etc.), new distributed data management techniques and systems (e.g. MapReduce, Hadoop, SciDB, Peanut, Pig latin, etc.) are emerging and receiving much attention from the research community. Although they do well in terms of consistency/flexibility/performance trade-offs for specific applications, they seem to be ad-hoc and might hurt data interoperability. The key questions I discuss are: What are the fundamental principles behind the emerging solutions? Is there any generic architectural model, to explain those principles? Do we need new foundations to look at data distribution?

1 Introduction

The 1980's were very active periods for the development of distributed relational database technology and all commercial DBMSs today are distributed. The decade of 1990's saw the development and maturation of client-server technology and the introduction of object-orientation – both as stand-alone systems and as object-relational DBMSs.

The Özsu-Valduriez textbook Principles of Distributed Database Systems [10] was first published in 1991, covering the fundamental distribution principles and techniques. The second edition was published in 1999 and included coverage of client-server systems and distributed object systems. The third edition of the book was out in April 2011. During the writing of the third edition, we have been evaluating the past and contemplating the future. It has been almost twenty years since the first edition appeared, and ten years since the second edition. As one can imagine, in a fast

* Work partially funded by the DataRing project of the French ANR.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 1–11, 2011.

changing area such as this, there have been significant changes in the intervening period. As we wrote the third edition, we incorporated technologies that were developed in late 1990's and in 2000's – P2P systems, data integration, database clusters, web and XML data management, stream data management, and cloud data management. It is apparent that the last ten years have seen an accelerated investigation of distributed data management technologies spurred by advent of high-speed networks, fast commodity hardware, very heavy parallelization of hardware, and, of course, the increasing pervasiveness of the web.

Now, the question is what is likely to happen in the next decade; or to put it differently, if there were to be a fourth edition of our book in 2020, *what would it be? what would be new?* This is the motivation for this paper[1].

In observing the changes that have taken place over the past twenty years of our involvement with this field, what has struck as interesting is that the fundamental principles of distributed data management still hold, and distributed data management can be characterized on three dimensions: distribution, heterogeneity and autonomy of the data sources. What has changed much since and made the problems much harder, is the scale of the dimensions: very large scale distribution (cluster, P2P, web and cloud); very high heterogeneity (web); and high autonomy (web, P2P). Also interesting to note is that the fundamental principles of database fragmentation (or partitioning), data integration, transaction management, replication and relational query processing have stood the test of time. In particular, new techniques and algorithms could be presented as extensions of earlier material, using relational concepts.

Today, to support the requirements of important data-intensive applications (e.g. social networks, web data analytics), new distributed data management techniques (e.g. MapReduce, Hadoop, Peanut, Pig latin, SciDB) are emerging and receiving much attention from the research community. Although they do well in terms of consistency-flexibility-performance trade-offs for specific applications, they seem to be ad-hoc and might hurt data interoperability. The key questions are: What are the fundamental principles behind the emerging solutions? Is there any generic architectural model, to explain those principles? Do we need new foundations to look at data distribution?

In this paper, I discuss these questions. I first recall the fundamental principles of distributed data management (Section 2). Then, in Section 3, I illustrate the challenges introduced by new data-intensive applications in the context of scientific applications and cloud computing. In Section 4, I discuss emerging solutions and show that they can still be explained along the three main dimensions of distributed data management (distribution, autonomy, heterogeneity). Finally, in Section 5, I discuss what is likely to come next.

2 Principles of Distributed Data Management

The fundamental principle behind data management is *data independence*, which enables applications and users to share data at a high conceptual level while ignoring implementation details. This principle has been achieved by *database systems* which

[1] This was also the basis for a panel at ICDE 2011 [11].

provide advanced capabilities such as schema management, high-level query languages, access control, automatic query processing and optimization, transactions, data structures for supporting complex objects, etc.

A *distributed database* is a collection of multiple, logically interrelated databases distributed over a computer network. A *distributed database system* is defined as the software system that permits the management of the distributed database and makes the distribution *transparent* to the users. Distribution transparency extends the principle of data independence so that distribution is not visible to users.

These definitions assume that each site logically consists of a single, independent computer. Therefore, each site has the capability to execute applications on its own. The sites are interconnected by a computer network with loose connection between sites which operate independently. Applications can then issue queries and transactions to the distributed database system which transforms them into local queries and local transactions (see Figure 1) and integrates the results. The distributed database system can run at any site s, not necessarily distinct from the data (i.e. it can be site 1 or 2 in Figure 1).

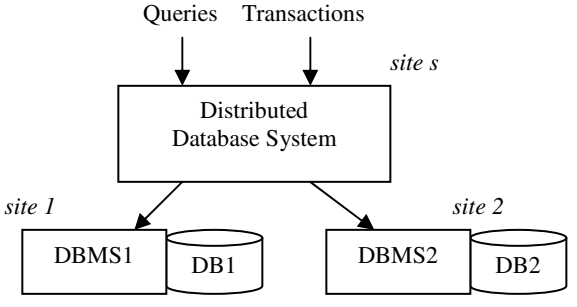


Fig. 1. A distributed database system with two data sites

The database is physically distributed across the data sites by fragmenting and replicating the data. Given a relational database schema, for instance, fragmentation subdivides each relation into partitions based on some function applied to some tuples' attributes. Based on the user access patterns, each of the fragments may also be replicated to improve locality of reference (and thus performance) and availability. The use of a set-oriented data model (like relational) has been crucial to define fragmentation, based on data subsets.

The functions provided by a distributed database system could be those of a database system (schema management, access control, query processing, transaction support, etc). But since they must deal with distribution, they are more complex to implement. Therefore, many systems support only a subset of these functions.

When the data and the databases already exist, one is faced with the problem of providing integrated access to heterogeneous data. This process is known as *data integration*: it consists in defining a *global schema* over the existing data and *mappings* between the global schema and the local database schemas. Data integration systems have received several names such as federated database systems, multidatabase systems and, more recently, mediator systems. Standard protocols such

as Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) ease data integration using SQL. In the context of the Web, mediator systems [13] allow general access to autonomous data sources (such as files, databases, documents, etc.) in read only mode. Thus, they typically do not support all database functions such as transactions and replication.

When the architectural assumption of each site being a (logically) single, independent computer is relaxed, one gets a *parallel database system* [14], i.e. a database system implemented on a tightly-coupled multiprocessor or a cluster. The main difference with a distributed database system is that there is a single operating system which eases implementation and the network is typically faster and more reliable. The objective of parallel database systems is high-performance and high-availability. High-performance (i.e. improving transaction throughput or query response time) is obtained by exploiting data partitioning and query parallelism while high-availability is obtained by exploiting replication. Again this has been made possible by the use of a set-oriented data model, which eases parallelism, in particular, independent parallelism between data subsets.

The distributed database approach has proved effective for applications that can benefit from static administration, centralized control and full-fledge database capabilities, e.g. information systems. For administrative reasons, the distributed database system typically runs on a separate server, which reduces scalability to tens of databases. Data integration systems achieve better scalability to hundreds or thousands of data sources by restricting functionality (i.e. read-only querying). Parallel database systems can also scale up to large configurations with thousands of processing nodes. However, both data integration systems and parallel database systems typically rely on a global schema that can be either centralized or replicated.

We now consider the possible ways in which a distributed DBMS may be architected. We use a classification (Figure 1) that organizes the systems as characterized with respect to three dimensions: (1) the autonomy of local systems, (2) their distribution, and (3) their heterogeneity. Autonomy, in this context, refers to the distribution of control, not of data. It indicates the degree to which individual DBMSs can operate independently. Whereas autonomy refers to the distribution (or decentralization) of control, the distribution dimension of the taxonomy deals with the physical distribution of data over multiple sites (or nodes in a parallel system). There are a number of ways DBMSs have been distributed. We distinguish between client/server (C/S) distribution and peer-to-peer (P2P) distribution (or full distribution). With C/S DBMS, sites may be clients or servers, thus with different functionality, whereas with homogeneous P2P DBMS (DDBMS in Figure 1), all sites provide the same functionality. Note that DDBMS came before C/S DBMS in the late 1970. P2P data management struck back in the 2000 with “modern” variations to deal with very large scale, autonomy and decentralized control. Heterogeneity refers to data models, query languages, and transaction management protocols. Multidatabase systems (MDBMS) deal with heterogeneity, in addition to autonomy and distribution.

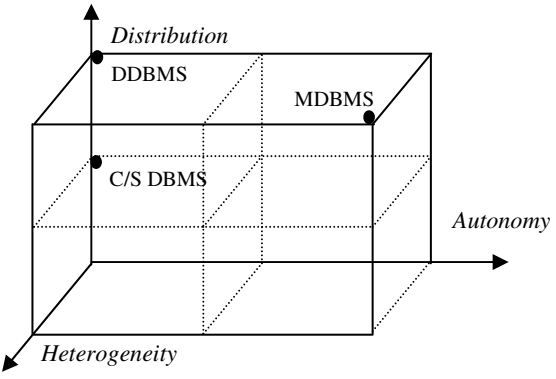


Fig. 2. Distributed DBMS Architectures (modified after [10])

3 New Challenges for Distributed Data Management

The pervasiveness of the web has spurred all kinds of data-intensive applications and introduced great challenges for distributed data management. New data-intensive applications such as social networks, web data analytics and scientific applications have requirements that are not met by the traditional distributed database systems in Figure 2. What has changed much and made the problems much harder, is the scale of the dimensions: very large scale distribution, very high heterogeneity, and high autonomy. Let us illustrate the challenges for distributed data management with two important domains: scientific data management and cloud data management.

3.1 Scientific Data Management

Scientific data management has become a major challenge for the database and data management research community [7]. Modern science such as agronomy, bio-informatics, physics and environmental science must deal with overwhelming amounts of experimental data produced through empirical observation and simulation. Such data must be processed (cleaned, transformed, analyzed) in all kinds of ways in order to draw new conclusions, prove scientific theories and produce knowledge. However, constant progress in scientific observational instruments (e.g. satellites, sensors, large hadron collider) and simulation tools (that foster in silico experimentation, as opposed to traditional in situ or in vivo experimentation) creates a huge data overload. For example, climate modeling data are growing so fast that they will lead to collections of hundreds of exabytes expected by 2020.

Scientific data is also very complex, in particular because of heterogeneous methods used for producing data, the uncertainty of captured data, the inherently multi-scale nature (spatial scale, temporal scale) of many sciences and the growing use of imaging (e.g. satellite images), resulting in data with hundreds of attributes, dimensions or descriptors. Processing and analyzing such massive sets of complex scientific data is therefore a major challenge since solutions must combine new data management techniques with large-scale parallelism in cluster, grid or cloud environments [12].

Furthermore, modern science research is a highly collaborative process, involving scientists from different disciplines (e.g. biologists, soil scientists, and geologists working on an environmental project), in some cases from different organizations distributed in different countries. Since each discipline or organization tends to produce and manage its own data, in specific formats, with its own processes, integrating distributed data and processes gets difficult as the amounts of heterogeneous data grow.

Despite their variety, we can identify common features of scientific data [1]: massive scale; manipulated through complex, distributed workflows; typically complex, e.g. multidimensional or graph-based; with uncertainty in the data values, e.g., to reflect data capture or observation; important metadata about experiments and their provenance; heavy floating-point computation; and mostly append-only (with rare updates).

3.2 Cloud Data Management

Cloud computing is the latest trend in distributed computing and has been the subject of much hype. The vision encompasses on demand, reliable services provided over the Internet (typically represented as a cloud) with easy access to virtually infinite computing, storage and networking resources. Through very simple Web interfaces and at small incremental cost, users can outsource complex tasks, such as data storage, system administration, or application deployment, to very large data centers operated by cloud providers. Thus, the complexity of managing the software/ hardware infrastructure gets shifted from the users' organization to the cloud provider. From a technical point of view, the grand challenge is to support in a cost-effective way the very large scale of the infrastructure which has to manage lots of users and resources with high quality of service.

However, not all data-intensive applications are good candidates for being "cloudified" [1]. To simplify, we can classify between the two main classes of data-intensive applications: On Line Transaction Processing (OLTP) and On Line Analytical Processing (OLAP). Let us recall their main characteristics. OLTP deals with operational databases of average sizes (up to a few Terabytes), is write-intensive, and requires complete ACID transactional properties, strong data protection and response time guarantees. On the other hand, OLAP deals with historical databases of very large sizes (up to Petabytes), is read-intensive and thus can accept relaxed ACID properties. Furthermore, since OLAP data are typically extracted from operational OLTP databases, sensitive data can be simply hidden for analysis (e.g. using anonymization) so that data protection is not as crucial as in OLTP.

OLAP is more suitable than OLTP for cloud primarily because of two cloud characteristics (see the detailed discussion in [1]): elasticity and security. To support elasticity in a cost-effective way, the best solution that most cloud providers adopt is a shared-nothing cluster. Shared-nothing provides high-scalability but requires careful data partitioning. Since OLAP databases are very large and mostly read-only, data partitioning and parallel query processing are effective. However, it is much harder to support OLTP on shared-nothing because of ACID guarantees which require complex concurrency control. For these reasons and because OLTP databases are not so large, shared-disk is the preferred architecture for OLTP.

The second reason that OLTP is not so suitable for cloud is that highly sensitive data get stored at an untrusted host (the provider site). Storing corporate data at an untrusted third-party, even with a carefully negociated Service Level Agreement (SLA) with a reliable provider, creates resistance from some customers because of security issues. However, this resistance is much reduced for historical data, with anonymized sensitive data.

There is much more variety in cloud data than in scientific data since there are many different kinds of customers (individuals, SME, large corporations, etc.). However, we can identify common features. Cloud data can be very large, unstructured (e.g. text-based) or semi-structured, and typically append-only (with rare updates). And cloud users and application developers may be in high numbers, but not DBMS experts.

4 Emerging Solutions

Generic data management solutions (e.g. relational DBMS) that have proved effective in many application domains (e.g. business transactions) are not efficient at dealing with these emerging applications, thereby forcing developers to build ad-hoc solutions that are labor-intensive and cannot scale. In particular, relational DBMSs have been lately criticized for their "one size fits all" approach. Although they have been able to integrate support for all kinds of data (e.g., multimedia objects, XML documents and new functions), this has resulted in a loss of performance and flexibility for applications with specific requirements because they provide both "too much" and "too little". Therefore, it has been argued that more specialized DBMS engines are needed. For instance, column-oriented DBMSs, which store column data together rather than rows in traditional row-oriented relational DBMSs, have been shown to perform more than an order of magnitude better on decision-support workloads. The "one size does not fit all" counter-argument generally applies to cloud data management as well.

Therefore, current data management solutions have traded consistency for scalability, simplicity and flexibility. As alternative to relational DBMS (which use the standard SQL language), these solutions have been recently quoted as Not Only SQL (NOSQL) by the database research community. Many of these solutions have been developed for the cloud or the grid, which both exploit large scale parallelism, typically with shared-nothing clusters.

Figure 3 positions the architectures of emerging solutions along the same three dimensions (distribution, heterogeneity and autonomy). Like cloud computing, grid computing enables access to very large compute and storage resources over the Web. Compared with cloud computing which deals with large-scale parallelism, the grid is characterized with high heterogeneity, large-scale distribution and large-scale parallelism. In addition to grid and cloud, we also position the recent P2P DBMS that target large-scale distribution and data integration systems (like MDBMS) that deal with large-scale distribution, heterogeneity and autonomy.

Distributed data management for cloud applications emphasizes scalability, fault-tolerance and availability, sometimes at the expense of consistency or ease of development. Let us illustrate this approach with two popular solutions: Google Bigtable and Yahoo! PNUTS.

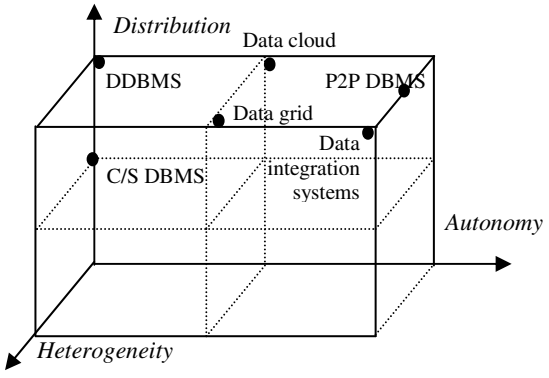


Fig. 3. Architectures of Emerging Solutions

Bigtable is a database storage system for a shared-nothing cluster [4]. It uses a distributed file system (Google File System - GFS) for storing structured data in distributed files, with fault-tolerance and availability. It also uses a form of dynamic data partitioning for scalability. There are also open source implementations of Bigtable, such as Hadoop Hbase, which runs on Hadoop Distributed File System (HDFS). Bigtable supports a simple data model that resembles the relational model, with multi-valued, timestamped attributes. It provides a basic API for defining and manipulating tables, within a programming language such as C++, and various operators to write and update values, and to iterate over subsets of data, produced by a scan operator. There are various ways to restrict the rows, columns and timestamps produced by a scan, as in a relational select operator. However, there is no complex operator such as join or union, which should be programmed using the scan operator. Transactional atomicity is supported for single row updates only. To store a table in GFS, Bigtable uses range partitioning on the row key. Each table is divided into partitions, called tablets, each corresponding to a row range.

PNUTS is a parallel and distributed database system for cloud applications at Yahoo! [5]. It is designed for serving Web applications, which typically do not need complex queries, but require good response time, scalability and high availability and can tolerate relaxed consistency guarantees for replicated data. PNUTS supports the relational data model, with arbitrary structures allowed within attributes of Blob type. Schemas are flexible as new attributes can be added at any time even though the table is being queried or updated, and records need not have values for all attributes. PNUTS provides a simple query language with selection and projection on a single relation. Updates and deletes must specify the primary key. PNUTS provides a replica consistency model that is between strong consistency and eventual consistency, with several API operations with different guarantees. Database tables are horizontally partitioned into tablets, through either range partitioning or hashing, which are distributed across many servers in a cluster (at a site).

To summarize, both Bigtable and PNUTS provide some variations of the relational model, a simple API or language for manipulating data, and relaxed consistency guarantees. They also rely on fragmentation (partitioning) and replication for fault-tolerance. Thus, they capitalize on the well-known principles of distributed data management.

Emerging solutions strive to be more generic than ad-hoc solutions which are labor-intensive and cannot scale. For instance, the SciDB organization (http://www.scidb.org) is building an open source database system for scientific data analytics. SciDB will be certainly effective for similar applications for which the data is well understood (with well-defined data structures). However, to avoid that the "one size fits all" argument applies to SciDB as well, the key question is: *how generic should scientific data management be, without hampering application-specific optimizations?* For instance, to perform scientific data analysis efficiently, scientists typically resort to dedicated indexes, compression techniques and specific algorithms. Thus, generic techniques, inspired from the DB research community should be able to cope with these specific techniques.

Genericity in data management encompasses two dimensions: data model (which provides data structures (captured by the data model) and data processing (inferred by the query language). Relational DBMS have initially provided genericity through the relational data model (that subsumes earlier data models) and a high-level query language (SQL). However, successive object extensions to include new data structures such as lists and arrays and support user-defined functions in a programming language have resulted in a yet generic, but more complex data model and language for the developers. Therefore, emerging solutions tend to rely on a more specific data model (e.g. Bigtable which is some kind of nested relational model) with a simple set of operators easy to use from a programming language. For instance, to address the requirements of social network applications, new solutions rely on a graph data model and graph-based operators. To address the requirements of scientific applications, SciDB supports an array data model, which generalizes the relational model, with array operators. User-defined functions also allow for more specific data processing. MapReduce [6] is a good example of generic parallel data processing framework, on top of a distributed file system (GFS). It supports a simple data model (sets of (key, value) pairs), which allows user-defined functions (map and reduce). Although quite successful among developers, it is relatively low-level and rigid, leading to custom user code that is hard to maintain and reuse. Pig latin [8] is an alternative data management solution that raises the level of abstraction with an algebraic query language. In emerging solutions, it is interesting to witness the development of algebras, with specific operators, to raise the level of abstraction in a way that enables optimization. For instance, in [9], we propose an algebraic approach enables automatic optimization of scientific workflows that manipulate huge amounts of data through specific programs and files.

5 Conclusion

To support the requirements of important data-intensive applications (e.g. social networks, web data analytics, scientific applications, etc.), new distributed data management techniques and systems (e.g. MapReduce, Hadoop, SciDB, Peanut, Pig latin, etc.) have emerged and are receiving much attention from the research community. Now, the guiding question for this paper was what is likely to happen in the next decade; or what will be the principles of distributed data management in 2020. I translated this into three questions: (1) What are the fundamental principles

behind the emerging solutions? (2) Is there any generic architectural model, to explain those principles? (3) Do we need new foundations to look at data distribution?

To address (1), I illustrated the challenges introduced by new data-intensive applications in the context of scientific applications and cloud computing. The emerging solutions typically provide some variations of the relational model, a simple API or language for manipulating data, and relaxed consistency guarantees. They also rely on fragmentation (partitioning) for large-scale parallelism and replication for fault-tolerance. Thus, they capitalize on the well-known principles of distributed data management.

Wrt (2), I showed that emerging solutions can still be explained along the three main dimensions of distributed data management (distribution, autonomy, heterogeneity), yet pushing the scales of the dimensions high up. However, I raised the question of how generic should distributed data management be, without hampering application-specific optimizations. Emerging NOSQL solutions tend to rely on a specific data model (e.g. Bigtable, MapReduce) with a simple set of operators easy to use from or with a programming language. It is also interesting to witness the development of algebras, with specific operators, to raise the level of abstraction in a way that enables optimization [9]. What is missing to explain the principles of emerging solutions is one or more dimensions on generic/specific data model and data processing.

The hardest question is (3): Do we need new foundations to look at data distribution? It all depends what we mean by "data" and whether we consider the continuum between data, information and knowledge, with increasing pervasiveness of data semantics. During the ICDE 2011 panel [11], one of us (S. Abiteboul) argued that accessing highly distributed, heterogeneous data in personal dataspaces on the web was beyond human expertise, and requires us moving from data to knowledge, with automated reasoning. This is the motivation for the comeback of Datalog as a uniform language to deal with data, metadata, rules, distribution, time, etc. [2].

Acknowledgements. This paper has benefited from fruitful discussions with many colleagues: Tamer M. T. Özsu, of course, during the writing of [10]; Serge Abiteboul, Bettina Kemme, Ricardo Jiménez-Peris, Beng Chin Ooi in the ICDE 2011 panel on the same topic [11]; the members of the Zenith team at INRIA-LIRMM, in particular, Reza Akbarinia, Florent Masseglia and Esther Pacitti; and the members of the CNPq-INRIA project Datluge, in particular, Marta Mattoso, Eduardo Osgarawa, and Fabio Porto.

References

1. Abadi, D.J.: Data Management in the Cloud: Limitations and Opportunities. IEEE Data Engineering Bulletin 32(1), 3–12 (2009)
2. Abiteboul, S., Bienvenu, M., Galland, A., Antoine, E.: A Rule-based Language for Web Data Management. In: ACM Symposium on Principles of Database Systems, PODS (2011)
3. Ailamaki, A., Kantere, V., Dash, D.: Managing scientific data. CACM 53(6), 68–78 (2010)
4. Chang, F., et al.: Bigtable: A Distributed Storage System for Structured Data. ACM TOCS 26(2) (2008)

5. Cooper, B.F., et al.: PNUTS: Yahoo!'s Hosted Data Serving Platform. Proceedings of the VLDB Endowment (PVLDB) 1(2), 1277–1288 (2008)
6. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Symp. on Operating System Design and Implementation (OSDI), pp. 137–150 (2004)
7. Gray, J., Liu, D.T., Nieto-Santisteban, M.A., Szalay, A.S., DeWitt, D.J., Heber, G.: Scientific Data Management in the Coming Decade. Technical Report MSR-TR-2005-10, Microsoft Research (2005)
8. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. In: ACM SIGMOD Int. Conf. on Management of Data (SIGMOD), pp. 1099–1110 (2008)
9. Osgarawa, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M.: An Algebraic Approach for Data-Centric Scientific Workflows. In: Proceedings of the VLDB Endowment, PVLDB (to appear, 2011)
10. Özsu, T., Valduriez, P.: Principles of Distributed Database Systems, 3rd edn. Springer, Heidelberg (2011)
11. Özsu, T., Valduriez, P., Abiteboul, S., Kemme, B., Jiménez-Peris, R., Chin Ooi, B.: Distributed data management in 2020? In: IEEE Int. Conf. On Data Engineering (ICDE), vol. 1360 (2011)
12. Pacitti, E., Valduriez, P., Mattoso, M.: Grid Data Management: open problems and new issues. Journal of Grid Computing 5(3), 273–281 (2007)
13. Tomasic, A., Raschid, L., Valduriez, P.: Scaling access to heterogeneous data sources with DISCO. IEEE Trans. on Knowledge and Data Engineering 10(5), 808–823 (1998)
14. Valduriez, P.: Parallel Database Systems: open problems and new issues. Int. Journal on Distributed and Parallel Databases 1(2), 137–165 (1993)
15. Valduriez, P., Pacitti, E.: Data Management in Large-Scale P2P Systems. In: Daydé, M., Dongarra, J., Hernández, V., Palma, J.M.L.M. (eds.) VECPAR 2004. LNCS, vol. 3402, pp. 104–118. Springer, Heidelberg (2005)

The Science of Conceptual Modelling

Bernhard Thalheim

Christian-Albrechts-University Kiel, Computer Science Institute, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de
http://www.is.informatik.uni-kiel.de/~thalheim

Abstract. Conceptual modelling is one of the central activities in Computer Science. Conceptual models are mainly used as intermediate artifact for system construction. They are schematic descriptions of a system, a theory, or a phenomenon of an origin thus forming a model. A conceptual model is a model enhanced by concepts. The process of conceptual modelling is ruled by the purpose of modelling and the models. It is based on a number of modelling acts, on a number of correctness conditions, on modelling principles and postulates, and on paradigms of the background or substance theories. Purposes determine the (surplus) value of a model. Conceptual modelling is performed by a modeller that directs the process based on his/her experience, education, understanding, intention and attitude. Conceptual models are products that are used by other stakeholders such as programmers, learners, business users, and evaluators. Conceptual models use a language as a carrier for the modelling artifact and are restricted by the expressiveness of this carrier.

This paper aims at a discussion of a general theory of *modelling as a culture and an art*. A general theory of modelling also considers modelling *as an apprenticeship* and *as a technology*. It is thus an *art*. Modelling is on of the main elements of Computer Science *culture* that consists of commonly accepted behaviour patterns, arts, consensus, institutions, and all other supporting means and thoughts.

1 The Triptychon of Model as an Artifact, Modelling as an Activity and Modelling as an Art and Science, Thus as a Culture

Conceptual modelling is a widely applied practice in Computer Science and has led to a large body of knowledge on constructs that might be used for modelling and on methods that might be useful for modelling. It is commonly accepted that database application development is based on conceptual modelling. It is however surprising that only very few publications have been published on a *theory of conceptual modelling*. We continue the approach [18,19] and aim in a theory of modelling within this paper. An approach to a theory of models has been developed in [19]. An approach to a theory of model activities is discussed in [18].

1.1 Differences between 'Model', 'To Model' and 'Modelling'

The conceptions of model, of the activity 'to model' and of modelling are often used as synonyms. We must however distinguish these conceptions for a theory of models,

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 12–26, 2011.

a theory of model activities and a theory of the modelling process. Similar to notions in philosophy of science [11] we distinguish between the conception of a model, the conception of a model activity, and the conception of modelling processes.

The model as an artifact: The model is something set or held for guidance or imitation of an origin and is a product at the same time. Models are enduring, justified and adequate artifacts from one side. From the other side, models represent the state of comprehension or knowledge of a user.

To model as an activity: 'To model' is a scientific or engineering activity beside theoretical or experimental investigation. The activity is an additive process. Corrections are possible during this activity. Modelled work may be used for construction of systems, for exploration of a system, for definition and negotiation, for communication, for understanding and for problem solving.

Modelling as a systematically performed technological process: Modelling is a technique of systematically using knowledge from computer science and engineering to introduce technological innovations into the planning and development stages of a system. At each stage the modeller is likely to ask both why and how, rather than merely how. Modelling is thus based on paradigms and principles.

Additionally, the notion of model may be used in an adjective sense as serving as or capable of serving as a pattern or being a usually miniature representation of something. This notion is often used for sample representations such as a 'model chair'. Another notion of the model that is not of interest within this paper is the miniature representation of something.

1.2 The Simultaneity of Art, Culture, Technology and Techniques in Modelling

Modelling can be understood as a technique[1] or as a technology[2]. [11] distinguishes between science and technology: Technology is the systematic study of techniques for making and doing things; science is the systematic attempt to understand and interpret the world. While technology is concerned with the fabrication and use of artifacts, science is devoted to the more conceptual enterprise of understanding the environment, and it depends upon the comparatively sophisticated skills of literacy and numeracy.

At the same time, modelling is an art[3]. Modelling is a highly creative process. It requires skills in planning, making, or executing. It is often claimed that it is not to be formalisable. It requires deep insight into the background as well as skills, careful

[1] I.e., the fashion, manner, mode, modus, system, way, wise in which a system etc. is mastered. Techniques consist of methods of accomplishing a desired aim.

[2] Technology is an element of engineering. It consists of the practical application of knowledge especially in a particular area. It provides a capability given by the practical application of knowledge. Therefore, it is a manner of accomplishing a task especially using technical processes, methods, or knowledge.

[3] Art requires capability, competence, handiness, and proficiency. Art is based on finesse, i.e. on refinement or delicacy of workmanship. Models and art share a Janus head evaluation: The judgement of beauty evaluates the model within a community of business users. The judgement of the sublime evaluates the model against its technical realisation. A model has thus both an extrinsic and intrinsic value.

simplification, experience and ingenuity. Due to the variety of viewpoints, modelling is also based on judgement and clever selection with different alternatives.

Modelling in one of the main activities in Computer Science. It consists of commonly accepted and practised behavior patterns, arts, consensus, institutions, and all other products of human work and thought. Turning to [111][4], culture is based on the capacity for rational or abstract thought. The meaning of abstraction is not sufficiently explicit or precise. The term symboling has been proposed as a more suitable name for assigning to things and events certain meanings that cannot be grasped with the senses alone.

This culture is learned and shared within communities which have their own behaviour pattern and approaches. It is not yet a science since it heuristically uses operational and/or scientific terms.

1.3 Orientation of This Paper

This paper explores modelling as an art and culture. We base the discussion on a theory of models and of modelling activities. We abstract therefore in this paper from micro-, meso-, macro-models or model suites used in many natural sciences or model suites [17], e.g., model ensembles used in UML or OWL. We do not yet consider modelling competency or MDA/D. We do not yet consider modelling competency. All notions used in this paper are based on [16].

The main goal of this paper is to show that modelling requires apprenticeship and technology. The orientation towards an expert mode can be reached if modelling is based on systematic development and if modelling is considered to be a craft of modelling activities. This approach shows that modelling incorporates design science in a wider sense as it has been considered in the literature.

We base our ideas on our observations on model developments for very large database schemata and very large database systems[5]. Such systems require a well organised modelling process. They must be evolution-prone and revision-prone. The paper

[4] The notion of culture combines at least eight facets: (1) cultivation, tillage; (2) the act of developing the intellectual and moral faculties especially by education; (3) expert care and training; (4) enlightenment and excellence of taste acquired by intellectual and aesthetic training; (5) acquaintance with and taste as distinguished from vocational and technical skills; (6) integrated pattern of human knowledge, belief, and behavior that depends upon man's capacity for learning and transmitting knowledge to succeeding generations; (7) the customary beliefs, social forms, and material traits of a group; and (8) the set of shared attitudes, values, goals, and practices that characterizes a company or corporation.
Culture requires different practices: education, enlightenment, erudition, learning from one side, gentility, manners, discrimination, taste from the other side, and sophistication, class, and elegance from a third side.

[5] Due to our involvement into the development and the service for the CASE workbenchs $(DB)^2$ and ID^2 we have collected a large number of real life applications, e.g., the SAP R/3 schema. Some of them have been really large or very large, i.e., consisting of more than 1.000 attribute, entity and relationship types. The largest schema in our database schema library contains of more than 19.000 entity and relationship types and more than 60.000 attribute types that needs to be considered as different.

concentrates thus one of the main workflows: *description of application worlds* followed by *prescription for system worlds* and *specification of systems*.

2 The World of Models

2.1 The World of Models

Models are *artifacts* selected by a *stakeholder* based on some stakeholder *judgement* or *perception* and governed by the *purpose*. Models can thus be characterised by *main dimensions*:

purpose ("wherefore") of models and modelling with the intentions, goals, aims, and tasks that are going to be solved by the model,

result of mapping ("whereof") with a description of the solution provided by the model, the characterisation of the problem, phenomena, construction or application domain through the model,

language ("wherewith") with a careful selection of the the carrier or cargo[7] that allows to express the solution, the specification of the world or the construction, and

value ("worthiness") of a model by explicit statement of the internal and external qualities, and the quality of use, e.g. explicit statement of invariance properties relating the model to its associated worlds or by preservation properties that are satisfied by the model in dependence on the associated worlds.

These four dimensions are driven by two *context dimensions*: the application domain dimension rules the scope and (explicit and implicit) disregard of the model; the user or stakeholder dimension governs the viewpoint, orientation and background of users involved. The mapping associates the origin and the artifact. As far as we are interested in modelling of information systems, we may use a (semi-)formal language for the artifact.

These main dimensions of models and modelling govern the model and the modelling acts. They are extended by secondary dimensions that are used to shape and to adapt the model: the artifact, the user, the context and the application domain dimensions. The mapping dimension is discussed in [18]. The value dimension can be described based on [6]. The purpose dimension is ruling and *governing* both the development of models and the application of models. This tight governance is caused by the main aim of a model: to provide a solution to a problem.

2.2 The Model as a Physical or Virtual Artifact

The main product of modelling and model activities is the model, i.e. an artifact that is considered to be worth for its purpose by the author. The model can, for instance, be used for the description of the world of origins or for the prescription of constructions. There are a number of explicit choices an author makes and that rule applications of models. Modelling of information systems

depends on the *abstraction layer*, e.g. requirements, specification, realisation or implementation layer,

depends on chosen *granularity and precision* of the work product itself,
depends on *resources* used for development of a model such as the language,
depends on *level of separation of concern* such as static/dynamic properties, local/global scope, facets,
depends on *quality properties of the input* , e.g. requirements, completeness, conciseness, coherence, understandability,
depends on decomposition of the work products in ensembles of sub-products, and
satisfies quality characteristics such as quality in use, internal quality, and external quality.

The task of model development is never completed (ta panta rhei ($\tau\alpha\ \pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$), 'the rivers flow'; narrative: everything flows). Models are changing artifacts due to changes imposed by

scope insight for conscious handling of restriction, capabilities, opportunities,
guiding rules for convenience, for completion, refinement, and extension,
development plans for partial delivery of models, partial usage and deployment,
theories supporting development of models,
quality characteristics for model completion, model evolution, model engineering, and
mapping styles for mapping models among abstraction layers.

2.3 The Purpose Dimension

The purpose dimension is *ruling* and *governing* the model, the development process and the application process because of the main reason for using a model is to provide a solution to a problem. Therefore the purpose is characterised by the solution to the problem provided by the model. We may distinguish a number of concerns such as

the impact of the model ("whereto") for a solution to a problem,
the insight into the origin's properties ("how") by giving details how the world is structured or should be structured and how the functionality can be described,
restrictions on applicability and validity ("when") of a model for some specific solutions, for the validity interval, and the lifespan of a model,
providing reasons for model value ("why") such as correctness, generality, usefulness, comprehensibility, and novelty, and
the description of functioning of a model ("for which reason") based on the model capacity.

The task of model development is never completed (ta panta rhei ($\tau\alpha\ \pi\alpha\nu\tau\alpha\ \rho\epsilon\iota$), 'the rivers flow', narrative: everything flows). Models are *evolving artifacts* due to changes imposed by

- scope insight for conscious handling of restriction, capabilities, opportunities,
- guiding rules for convenience, for completion, refinement, and extension,
- development plans for partial delivery of models, partial usage and deployment,
- theories supporting development of models,
- quality characteristics for model completion, evolution and engineering, and
- mappings styles for mapping models among abstraction layers.

2.4 The Language Dimension

Models are represented by *artifacts* that satisfy the pragmatic purposes of users. In this case, artifacts are linguistic expressions that describe the model. Linguistic expressions are built within a language with some understanding. Therefore, artifacts use syntax, semantics and pragmatics built within the chosen language.

Models are often expressed through expressions in a formal language $\mathcal{L}_{\tilde{M}}$. A model should support its objectives. Optimally, these objectives $\Psi(M)$ can be expressed in the same language $\mathcal{L}_{\tilde{M}}$ that is also used for the model M. A model has a number of properties. Some of them are of interest and used for characterisation of the model, e.g., $\Phi(M)$. This characterisation depends on the model and its purpose.

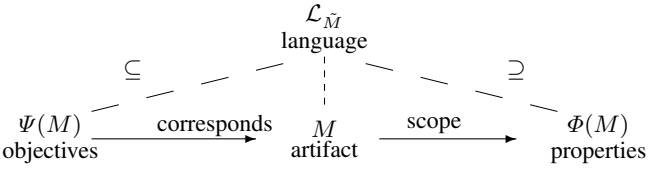


Fig. 1. Artifacts with a language, their properties and objectives within a given language for the artifact

Constructive languages are a special case and support

- the prescription of the objectives or postulates that restrict the judgement that an artifact can be accepted as a model,
- the scope of our attention to those artifacts that can be considered for a model or for parts of a model, and
- the orientation of the user on certain properties that are of interest for the purpose of modelling.

Natural languages have a high potential for deployment of deep semantics and cause a threat to everybody who does not use the language within the same semantical culture. Culture depends on participating stakeholders, their profile (educational, employment, psychological) and includes language, styles of communication, practices, customs, and views on roles and relationships. Deployment of natural language expressions may thus result in misunderstandings. There are two ways to avoid such: the development of a sophisticated ontology that includes all namespaces a user might use or the development of an *orthonormalised language* [9] that is restricted in expressivity and does not allow misinterpretations.

2.5 The Context Dimensions

The User Dimension. A number of users are involved into the development of models. The user dimension thus reflects intentions, the understanding, the comprehension and other characteristics of users in a variety of roles, e.g.,

the role of an *author ("by whom")* that results in reflections of the educational level, application of templates, pattern or reference models,
the role of an *addressee ("to whom")* that restricts the utilisation of the model or that supports the extended application beyond the purpose originally intended, and
the role of *broad public ("whichever")* that develops a common understanding of the model depending on the group or the culture of the public.

The Application Domain Dimension and the World of Origins. The application domain consists of people, organisational systems, and technical systems that interact to work towards a goal. This dimension clarifies

the *domain depending on models purpose ("for what")* such as an application domain, properties reflected or neglected,
the *scope to specific elements ("what")* that are considered to be typical and whose properties should be reflected,
the *attention within the domain depending on models purpose ("where")* that limits the model to the 'normal' aspects,
the *orientation of the domain ("wherefrom")* that restricts the attention and the issues for the current activities supported by the model,
the sources for origins or the infrastructure ("whence") considered for the model, and
the *restrictions of the world ("wherein")* associated with the model.

3 The World of Modelling Activities

3.1 Workflows Applied in the Model Development and Deployment Process

The purpose dimension governs the workflows applied in conceptual modelling. It also governs the kind of model application. We may distinguish a number of workflows in conceptual modelling such as the following ones:

Construction workflows are based on creation of models (as images, representations or portraits of the origin) that are used for production of systems (using as models as groundwork, background, pattern, standards, prototypes for the system). This kind of model exploitation uses the *dichotomy of models* as image of an origin and groundwork for a system.
Explanation workflows result in new insights into the world of the origins.
Optimisation-variation workflows result in an improvement and adaptation of the origins.
Verification-validation-testing workflows result in an improvement of the one of the subject, in most cases in an improvement of models.
Reflection-optimisation workflows are typical for mathematical modelling of the world of origins.
Explorative workflows are using models for learning about origins.
Hypothetical workflows are typical for discovery sciences, e.g., sciences used for climate research.
Documentation-visualisation workflows target on better understanding and comprehension of models.

These workflows can be intertwined or shuffled with each other. They may be performed one after another. In this paper we concentrate on the *construction* (or *creation-production*) workflow which seems to be central for information systems.

3.2 Modelling Acts

It surprises that these model activities are not explicitly handled in most modelling approaches. The same observation can be observed for a declaration of the main goals of the modelling act. Main modelling acts which are the following ones:

construct a model, a part of the model, a concept or a judgement, etc. (describe, delineate, fabricate, master),
communicate the judgements, the observations, the concepts, etc. (explain, express, verbalise or display),
understand the application domain, the system opportunities, etc. (cognise, identify, recognise, percept),
discover the problems, the potential, the solutions, etc. (interact, identify),
indicate properties of importance, relevance, significance, etc. (visualise, measure, suggest, inform),
variate and *optimise* a solution, a judgement, a concept, a representation depending on some criteria,
verify or *validate* or *test* a model, a solution, a judgement, a representation or parts of those,
control the scope of modelling, the styles or pattern, parts of a model, judgements, etc. (rule, govern, proofread, confirm, restrain, administer, arrange, stratify, standardise),
alternate or *compensate* or *replace* or *substitute* or *surrogate* models or parts of them, judgements, concepts, etc. (transfer, reassign, evolve, migrate, balance, correct, novate, truncate, ersatz).

The first and last four goals lead to a *datalogical* model that is structured according to technology. The other goals result in an *infological* model that is delivered to the needs of the user. We thus use a different frame of reference. The application of the results may thus be descriptive or prescriptive, constitutive or prognosticating, categorical or exegetic or contemplative or formulaic.

3.3 Models Serving Both as a Description of an Application (Domain and Problem) and as a Prescription for Construction (of Systems)

By taking a leaf out of D. Bjorner [1] book we divide information systems engineering into five main phases: (1) *application domain description* with properties that are of interest and that are of relevance, (2) requirements or *objectives prescription for a model*, (3) *model development* with a statement of properties that are obeyed by the model, (4) requirements of *objectives prescription for the construction* of an information system, and (5) *information systems construction* and coding with properties that are obeyed by the information system. Therefore, a model is used as a mediator between the application world and the systems world. The (application, model, system)-triple is reflected by the *information system development triptych* consisting of description of application world, prescription for construction and specification of systems.

Conceptualisation is an orthogonal phase that aims at a theoretical underpinning of models. It is used for *semantification* of models and for improvement of comprehensibility of models and explicit reasoning on elements used in models.

The application domain description is mapped to a model describing the application domain, its entities, functions, events, and behaviour. It is based on a formal, semi-formal or natural language which allows to formulate a set of theorems or postulates or properties that are claimed to behold of the domain model. The information system itself is an artifact too. The model mediates between this final artifact and the application. Models describe the problem to be solved for the application and which are used as starting point for implementation. They are also used for documentation of the system, for migration and evolution processes, for optimisation of systems, for control of parts of systems, and for simulation of systems. Models must reflect the structure of a system, the functionality of a system, the support facilities of a system and the collaboration environment of a system.

Therefore we concentrate on one of the workflows: the prescription of systems imposed by the description of an application domain and of the problems under solution. This workflow is often considered to be one of the main workflows. We may also use other workflows. The construction workflow is however a typical example of an engineering workflow[6]. Engineering is nowadays performed in a systematic and well-understood form.

3.4 The Construction Workflow Based on Information Systems Models

Modelling is based on an evolutionary process and thus consists of at least three sub-processes:

- *selection* including rigorous testing against the origin,
- *communication* for generation of a common understanding and a productive way of thinking within a community, and
- *accumulation* of results and integration of these results into future developments.

The construction workflow is one of the most prominent workflows in information systems modelling. Methodologies developed for software engineering can be directly applied to this workflow. They are however mainly oriented towards system construction. The systems description dimension is not as well explored. The combination of these two sub-workflows is shown in Figure 2. We need to include into this combination also the quality dimension. The body of knowledge of software engineering includes also a large set of quality characteristics. [6] develops an approach to systematic quality development. We integrate this systematic quality management.

[6] The difference between scientific exploration and engineering is characterised by [12] as follows: “Scientists look at things that are and ask ‘why’; engineers dream of things that never were and ask ‘why not’. Engineers use materials, whose properties they do not properly understand, to form them into shapes, whose geometries they cannot properly analyse, to resist forces they cannot properly assess, in such a way that the public at large has no reason to suspect the extent of their ignorance.” Modelling incorporates both engineering and science. It is thus considered to be an engineering science.

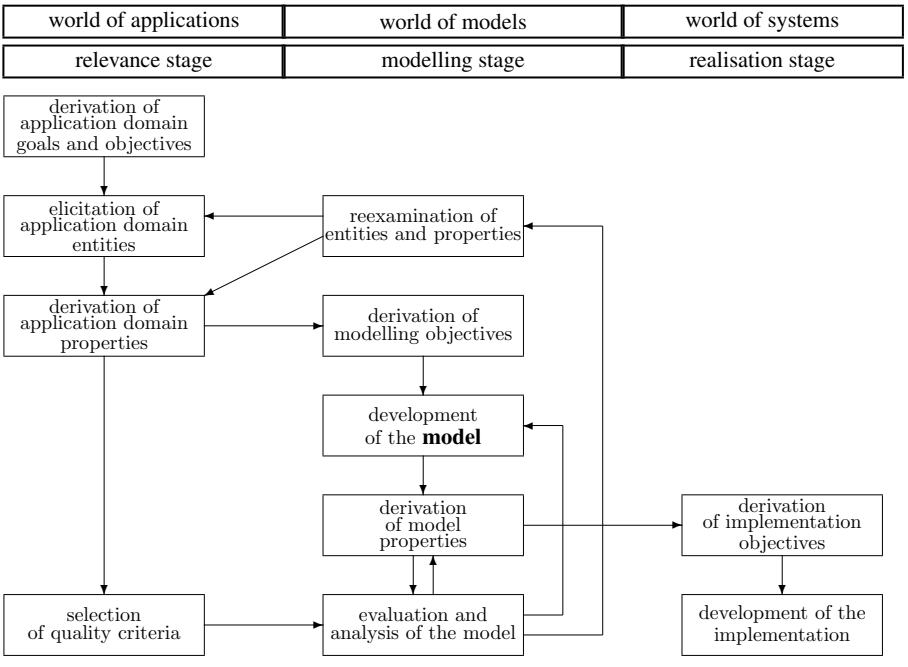


Fig. 2. The construction workflow that includes quality assurance

We can also develop other workflows such as agile modelling, spiral modelling and incremental modelling workflows. We restrict our attention in the sequel to the workflow in Figure 2 due to the length of this paper. This workflow separates three different worlds: the world of applications, e.g., the application domain in dependence on the purpose; the world of models, e.g. conceptual models used for information systems development; the world of systems, e.g., information systems combined with presentation systems. Based on this separation we can distinguish three stages: the relevance, modelling and realisation stages. This workflow reflects the separation into objectives, the artifact and the properties within the language dimension.

4 Duties and the Task Spectrum in Conceptual Modelling

4.1 The Design Science Approach to Tasks in Conceptual Modelling

MIS design science aims at the development of a general theory for models[7], model activities and modelling. We shall use the approach for a deeper insight into modelling. The management information system community characterises the modelling process by seven guidelines [3]:

[7] Models are called 'design' in [3].

(1) model are purposeful IT artifacts created to address a problem;
(2) models are solutions to relevant and important problems;
(3) the utility, quality, and efficacy of models must be evaluated by quality assessment;
(4) modelling research must contribute to the state of the art;
(5) modelling research relies upon the application of rigorous methods;
(6) modelling is a search process and use termination conditions;
(7) models must be communicated both to technology-oriented as well as to management audiences.

We observe that guidelines (1), (2), and (7) are characterising the model. Guidelines (3), (6) characterise model activities. Guideline (3), (5) is related to modelling as a technology. Guideline (4) is a general statement that relates modelling to a science.

Design science separates three cycles [20]: the relevance (or description) cycle, the design (or modelling) cycle, and the rigor (or conceptualisation) cycle.

4.2 CMM and SPICE for Conceptual Modelling

A software process is considered to be the set of activities, methods, and practices used in the production and evolution of software [4] and the associated products [10]. For improving of a software process there are four main approaches: modelling, assessment, measurement, and technology adoption [13]. The approaches supplement each other, but one of them is usually in a dominating position. CMMI and SPICE (Software Process Improvement and Capability dEtermination) [5] are the two most widely used software assessment models in software process improvement work today. The capability dimension consists of six *capability levels*: incomplete, performed, managed, established, predictable and optimizing.

Information system development is a specific software development process. Therefore, the SPICE characterisations are applicable as well. We may therefore distinguish different levels of the IS development *capability*:

1. *Performed and executed*: The goals of the application domain are satisfied. The information system development process is set out.
2. *Managed and defined*: Additionally, the application domain and scope are imaged by a model that allows to derive components of the system by means of model elements.
3. *Established and controlled*: The model is well-documented and allows to understand its design decisions. The model is used as a background and groundwork for the system.
4. *Understood, predictable and performed with sense*: The elements of the model are based on concepts that describe their semantics and meaning. The impact of languages as a model carrier, the assumptions made during design, the paradigms used during and the scope of the model are given in an explicit form.
5. *Optimised*: The model is developed with a number of alternatives. There are quantitative methods that support reasoning on quality of the model. Model alternatives can be given in a form that is the most adequate for the auditory. They can be used for deriving the best realisation.

4.3 The Duty Portfolio of Modelling

Following [2] we distinguish four main duties in conceptual modelling:

(1) Description: The application domain is described in a way that allows to comprehend the actual state, the necessities for system development and deployment, and the specifics and phenomena of the application.

(2) Explanation: The understanding of reality, of the processes and data in the application world and of the context of the application supports the creation of systems that effectively and efficiently support users. This understanding can be based on explication of concepts behind the application. It can also be based on behavioural pattern, on general laws and regulations and on user profiles and stories [14,15].

(3) Creation: The system creation includes coding of the system, embedding the system into a systems context, developing supporting means for users, and supporting a new behaviour of users of a system. It uses the demands stated for the application, the analysis of the current state, and the requests for change by the system. Creation includes elements of SWOT analysis (*s*trengthes, *w*eaknesses, *o*pportunities, *t*hreats) and evaluation of the quality of the system.

(4) Prognosis: The behaviour of the augmented system, the opportunities of changes and evolution and the restrictions of the augmented reality are predicted. The user expectations and the reality of system exploitation are compared on the basis of main storyboards observed for the applications.

We may now combine these approaches into a process survey in Figure 3.

The relevance cycle is based on observation of the state of affairs, scoping of the demands for system development, and describing a view of the application domain. These cycles form the y-dimension. We also use the x-dimension for explicit display of the changes imposed to the reality. Typically, information systems augment the reality. Figure 3 combines the approaches of design science (research) [3,8] with those based on main duties for system development [2] and those typically used for conceptual modelling [16].

The design or modelling cycle uses the scoped application domain description for the development of a model. The rigor cycle adds semantics, meaning and context to the model. The description of the scoped application domain may directly be used for system development. For instance, agile development is typically following this direct approach. The model may also be directly used for system development. The advantage of such approach is that all relevant elements are supported by a model and that the model may be used for understanding the system. The system is therefore defined. We may also use the model for development of a behaviour description, guidelines (e.g., for system deployment) and documentation. In this case modelling is established.

Furthermore, we might background the model by concepts. In this case, users of the model may perform system construction with a sense of groundwork behind the model and the description of the application domain. Models may also be a part of a knowledge base. In this case we integrate, generalise and found the model through concepts in the knowledge base.

The relevance, design and rigor cycles are based on comprehension of the application domain, perception of the relevant elements and knowledge or understanding development for those elements. During system development models are used as a mediating

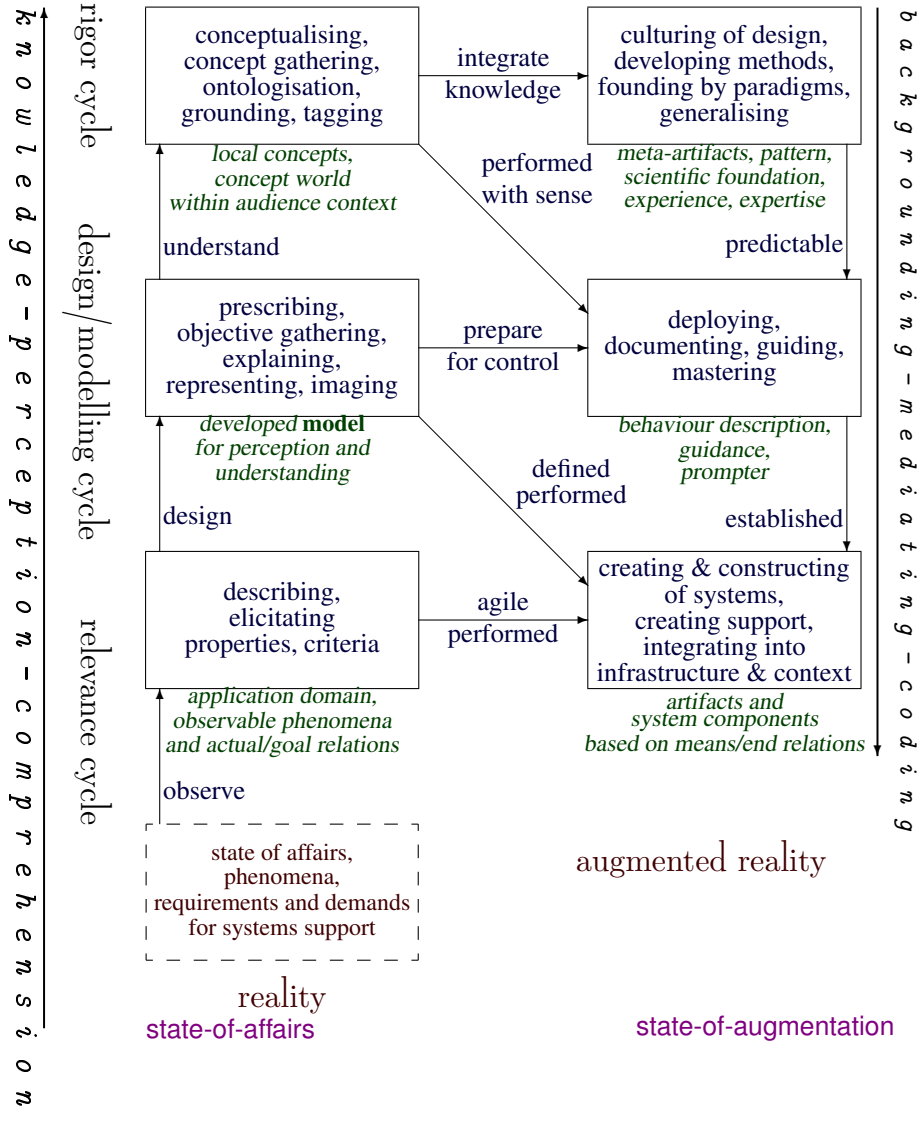


Fig. 3. The relevance/design/rigor and the state-of-affairs/augmentation dimensions

artifact. They describe and image the problems, phenomena and demand form one side and serve as a prescription for systems development from the other side. Models may also serve as a background and foundation of the system if they are integrated with concepts.

5 Conclusion

Models are artifacts that can be specified within a (W^4+$W^{17}H$)-frame based on the classical rhetorical frame introduced by Hermagoras of Temnos[8].

1. They are primarily characterised by W^4: wherefore (purpose), whereof (origin), wherewith (carrier, e.g., language), and worthiness ((surplus) value).
2. Secondary characterisation $W^{17}H$ is given by:
   - user or stakeholder characteristics: by whom, to whom, whichever;
   - characteristics imposed by the application domain: wherein, where, for what, wherefrom, whence, what;
   - purpose characteristics characterising the solution: how, why, whereto, when, for which reason; and
   - additional context characteristics: whereat, whereabout, whither, when.

Modelling combines at the same time and systematically different aspects: culture, art, systematics and technology of model (re)development and model application. It uses modelling activities and techniques.

Conceptual modelling is biased through a pragmatical culture. It uses languages as a sophisticated medium of expression. It defines its specific arts and sciences. It reflects thoughts, i.e. perception, interpretation and understanding of people involved. It is implicitly based on value systems transmitted through communities of practice based on some commonsense and consensus. Conceptual modelling is also at the same time a social activity, i.e. a shared pursuit within a community, demonstrated in a variety of textbooks, publications and conferences. Conceptual models are used for social aspects, i.e. include the give-and-take of socialisation, negotiation, protocol, and conventions within the community of their users. These aspects of models and of modelling activities collectively redefine conceptual modelling as a culture.

References

1. Bjørner, D.: Software Engineering 3: Domains, requirements, and software design. Springer, Berlin (2006)
2. Heinrich, L.J., Heinzl, A., Riedl, R.: Wirtschaftsinformatik: Einführung und Grundlegung, 4th edn. Springer, Berlin (2011)
3. Hevner, A., March, S., Park, J., Ram, S.: Design science in information systems research. MIS Quaterly 28(1), 75–105 (2004)
4. Humphrey, W.S.: Managing the Software Process. Addison-Wesley, Reading (1989)
5. ISO/IEC. Information technology - process assessment - part 2: Performing an assessment. IS 15504-2:2003 (2003)
6. Jaakkola, H., Thalheim, B.: Framework for high-quality software design and development: a systematic approach. IET Software 4(2), 105–118 (2010)
7. Mahr, B.: Information science and the logic of models. Softw. Syst. Model. 8, 365–383 (2009)

[8] Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis (W^7: Who, what, when, where, why, in what way, by what means). The Zachman frame uses a simplification of this frame.

8. March, S.T., Storey, V.C.: Design science in the information systems discipline: An introduction to the special issue on design science research. MIS Quarterly 4, 725–730 (2008)
9. Ortner, E., Schienmann, B.: Normative language approach - a framework for understanding. In: Thalheim, B. (ed.) ER 1996. LNCS, vol. 1157, pp. 261–276. Springer, Heidelberg (1996)
10. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability maturity model for software, version 1.1. Technical Report CMU/SEI-93-TR-024, Software Engineering Institute (February 1993)
11. Safra, J.E., Yeshua, I., et al.: Encyclopædia Britannica. Merriam-Webster (2003)
12. Samuel, A., Weir, J.: Introduction to Engineering: Modelling, Synthesis and Problem Solving Strategies. Elsevier, Amsterdam (2000)
13. Saukkonen, S., Oivo, M.: Six step software process improvement method (in finnish; teollinen ohjelmistoprosessi. ohjelmistoprosessin parantaminen SIPI-menetelmällä). Tekes 64/98, Teknologiakatsaus (October 1998)
14. Schewe, K.-D., Thalheim, B.: Reasoning about web information systems using story algebra. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, pp. 54–66. Springer, Heidelberg (2004)
15. Schewe, K.-D., Thalheim, B.: Usage-based storyboarding for web information systems. Technical Report 2006-13, Christian Albrechts University Kiel, Institute of Computer Science and Applied Mathematics, Kiel (2006)
16. Thalheim, B.: Entity-relationship modeling – Foundations of database technology. Springer, Berlin (2000)
17. Thalheim, B.: Model suites for multi-layered database modelling. In: Information Modelling and Knowledge Bases XXI. Frontiers in Artificial Intelligence and Applications, vol. 206, pp. 116–134. IOS Press, Amsterdam (2010)
18. Thalheim, B.: Towards a theory of conceptual modelling. Journal of Universal Computer Science 16(20), 3102–3137 (2010), http://www.jucs.org/jucs_16_20/towards_a_theory_of
19. Thalheim, B.: The theory of conceptual models, the theory of conceptual modelling and foundations of conceptual modelling. In: The Handbook of Conceptual Modeling: Its Usage and Its Challenges, ch. 17, pp. 547–580. Springer, Berlin (2011)
20. Venable, J.R.: Design science research post hevner et al.: Criteria, standards, guidelines, and expectations. In: Winter, R., Zhao, J.L., Aier, S. (eds.) DESRIST 2010. LNCS, vol. 6105, pp. 109–123. Springer, Heidelberg (2010)

Probabilistic Logics in Expert Systems: Approaches, Implementations, and Applications*

Gabriele Kern-Isberner[1], Christoph Beierle[2], Marc Finthammer[2], and Matthias Thimm[1]

[1] Dept. of Computer Science, TU Dortmund, 44221 Dortmund, Germany
[2] Dept. of Computer Science, FernUniversität in Hagen, 58084 Hagen, Germany

Abstract. The handling of uncertain information is of crucial importance for the success of expert systems. This paper gives an overview on logic-based approaches to probabilistic reasoning and goes into more details about recent developments for relational, respectively first-order, probabilistic methods like Markov logic networks, and Bayesian logic programs. In particular, we will feature the maximum entropy approach as a powerful and elegant method that combines convenience with respect to knowledge representation with excellent inference properties. We briefly describe some systems for probabilistic reasoning, and go into more details on the KReator system as a versatile toolbox for probabilistic relational learning, modelling, and inference. Moreover, we will illustrate applications of probabilistic logics in various scenarios.

1 Introduction

Real world applications of expert systems (and other computational systems, too) usually have to struggle with the problem that both background knowledge and information on the situation at hand are neither complete nor certain. For instance, in a medical domain, the physician may know that most patients suffering from appendicitis also complain about abdominal pain, but in some cases, the patients show other atypical symptoms; however, these relationships can not be further specified in a satisfactory way. In the special case of the patient she is just facing, she is not even sure whether he feels abdominal pain as he is a boy of three years of age.

Probabilistic logics offer a rich framework to represent and process uncertain information, and are linked to statistics and machine learning in a natural way. Knowledge can be extracted from data, expressed in a suitable probabilistic formalism like Bayesian networks [26], and used for uncertain reasoning by applying inference mechanisms. Completeness of knowledge can be achieved by presupposing additional assumptions like conditional independence of variables, like in most probabilistic networks [26], or by making use of the information-theoretical principles of optimum entropy [16]. In both ways, a full probability distribution

* The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grants KE 1413/2-2 and BE 1700/7-2).

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 27–46, 2011.

is generated from partial knowledge, on the base of which probabilities for arbitrary queries can be computed.

Most of the standard probabilistic approaches applied today make use of some type of probabilistic networks and propositional logic. While network techniques are of major importance to allow for local computations, the restriction to propositional logic makes probabilistic knowledge representation inadequate for domains in which relationships between objects are in the focus of investigation. So, especially in the last decade, a multitude of probabilistic approaches based on first-order logic have been brought forth. Markov logic networks [5], Bayesian logic programs [19], and similar relational probabilistic approaches [11] aim at generalising established propositional probabilistic methods to first-order knowledge representation. This turns out to be not an easy task, as the complexity of knowledge representation raises substantially, so that new inference techniques have to be devised. Moreover, the probabilistic semantics of open formulas is not at all clear. For example, the following conditional probabilistic formulas express commonsense knowledge about the relationships between elephants and their keepers which are usually good (elephants like their keepers), but also take exceptions into regard – elephants tend not to like keeper *fred*, except for the good natured elephant *clyde*:

$$(likes(X, Y) \mid elephant(X),\ keeper(Y))\ [0.8]$$
$$(likes(X, fred) \mid elephant(X))\ [0.3]$$
$$likes(clyde, fred)[0.9]$$

A schematic grounding of all rules of this knowledge base would cause conflicts with respect to elephant *clyde* and keeper *fred*. Moreover, both statistical (or population-based, respectively) information and subjective views are addressed, as the first two formulas involve all elephants (and keepers), while the third one only considers situations involving *clyde* and *fred*.

With many interesting new applications like social networks, hard computational problems, and challenging theoretical questions, the area of relational probabilistic knowledge representation has witnessed very active research work recently, providing a lot of new approaches and techniques. However, comparisons between approaches and evaluations with respect to computational and representational issues are not easily done, since each approach uses its own logical framework. Hence, there is an increasing demand for tools that support the investigation of different approaches in example or real world scenarios.

This paper aims to give an overview on the area of probabilistic knowledge representation, starting with standard propositional approaches and introducing into relational probabilistic knowledge representation by sketching some major approaches. As a special focus of the paper, we feature approaches that are based on the principle of maximum entropy as an elegant and powerful methodology that provides an excellent framework for commonsense and uncertain reasoning. Suitable systems are described briefly, and their use for applications in medical and biochemical domains is illustrated.

This paper is organized as follows. Sec. 2 provides details on Markov and Bayesian networks, and on the maximum entropy approach, both in propositional and first-order frameworks. In Sec. 3, the systems SPIRIT, MECORE, Alchemy, ProbCog, and KREATOR are presented, and applications of these systems are illustrated in Sec. 4. Finally, Sec. 5 concludes this paper.

2 Approaches

According to these approaches, Popular propositional approaches to probabilistic logic use probabilistic networks, heavily relying on the notion of conditional independence. According to these approaches, Probabilistic conditional logic employs the principle of maximum entropy and is also based on propositional logic, while Bayesian logic programs, Markov logic networks, and relational maximum entropy approaches support a relational setting.

2.1 Propositional Approaches

Probabilistic Networks. Conditional probabilities are often used in knowledge representation to describe causal or diagnostic dependencies [25]. A well-known framework which relies heavily on the notions of conditional probability and conditional independence are Bayesian networks. A Bayesian network BN for a set of propositions A is a tuple $BN = (A, E, P)$ such that (A, E) is a directed acyclic graph and P is a probability function that obeys

$$\{a\} \perp\!\!\!\perp_{\mathrm{P}} \mathsf{nd}(a) \,|\, \mathsf{pa}(a) \quad \text{(for every } a \in A\text{)}, \tag{1}$$

expressing that each vertex a is conditionally independent of its non-descendants $\mathsf{nd}(a)$, given the values of its parents $\mathsf{pa}(a)$. Condition (1) is also called the *local Markov property*. Due to this property, the probability function P can be decomposed into conditional probability functions for each node $a \in A$.

Example 1. We adapt an example on medical diagnosis, cf. [25]. Consider the propositions $A = \{a, b, c, d, e\}$ with the informal interpretations

a	cancer	b	increased serum calcium level
c	brain tumor	d	coma
e	headache		

and a Bayesian network $BN_{\mathrm{med}} = (A, E, P)$ with (A, E) given as depicted in Fig. 1. It follows that P has to adhere to the conditional independence $\{b\} \perp\!\!\!\perp_{\mathrm{P}} \{c\} \,|\, \{a\}$ (among others). Moreover, the probability of a possible world such as $ab\overline{c}\overline{d}e$ can be written as

$$P(ab\overline{c}\overline{d}e) = P(e \,|\, \overline{c}) \cdot P(\overline{d} \,|\, b\overline{c}) \cdot P(\overline{c} \,|\, a) \cdot P(b \,|\, a) \cdot P(a).$$

Therefore, P can be completely described by e. g. the following assignments[1]:

$$\begin{aligned} P(a) &= 0.20 \\ P(b \mid a) &= 0.80 & P(b \mid \overline{a}) &= 0.20 \\ P(c \mid a) &= 0.20 & P(c \mid \overline{a}) &= 0.05 \\ P(e \mid c) &= 0.80 & P(e \mid \overline{c}) &= 0.60 \\ P(d \mid b \wedge c) &= 0.80 & P(d \mid b \wedge \overline{c}) &= 0.90 \\ P(d \mid \overline{b} \wedge c) &= 0.70 & P(d \mid \overline{b} \wedge \overline{c}) &= 0.05 \end{aligned}$$

Note that the probabilities of negated variables derive from the above equations via e. g. $P(\overline{e} \mid c) = 1 - P(e \mid c)$. By only defining the above conditional probabilities the function P can be compactly stored.

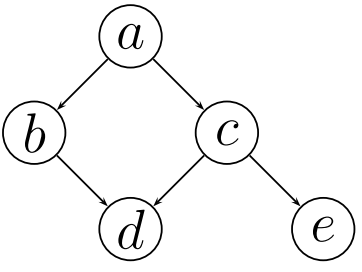


Fig. 1. The graph (A, E) from Ex. 1

Probabilistic Conditional Logic and Maximum Entropy. Conditional logic [22] is a knowledge representation formalism that concentrates on the role of *conditionals* or *if-then-rules*. A conditional of the form $(\psi \mid \phi)$ connects some detached pieces of information ϕ, ψ and represents a rule "*If* ϕ *then (usually, probably)* ψ". A *probabilistic conditional* is an expression of the form $(\psi \mid \phi)[d]$ with propositional formulas ϕ and ψ and $d \in [0, 1]$.

Example 2. The well-known penguin example that illustrates the problem of exceptions in subclasses can be represented as a knowledge base $\mathcal{R}$ with $\mathcal{R} = \{r_1, r_2, r_3\}$ with

$$r_1 = (bird \mid peng)[1.0] \qquad r_2 = (fly \mid bird)[0.9] \qquad r_3 = (fly \mid peng)[0.01].$$

A probability function P satisfies a probabilistic conditional

$$P \models (\psi \mid \phi)[d] \quad \text{if and only if} \quad P(\psi \mid \phi) = d \text{ and } P(\phi) > 0. \tag{2}$$

Reasoning in probabilistic conditional logic can be performed by maximizing entropy. The entropy $H(P)$ of a probability function P is defined via $H(P) = -\sum_\omega P(\omega) \log P(\omega)$ with $0 \cdot \log 0 = 0$. By selecting $P^* = \arg\max_{P \models \mathcal{R}} H(P)$

[1] The numbers have been arbitrarily chosen and may not describe the real world.

as the (unique) model of the knowledge base $\mathcal{R}$ with maximal entropy, one obtains a probability function that both satisfies all conditionals in $\mathcal{R}$ and adds as few additional information (in the information-theoretic sense) as possible. For a consistent knowledge base $\mathcal{R}$ the maximum entropy model P^* is uniquely determined, cf. [16]. Model-based probabilistic inference via P^* shows excellent logical properties [16], and has been proved to be most adequate for commonsense reasoning [24].

2.2 Relational Approaches

Bayesian Logic Programs. *Bayesian logic programming* combines logic programming and Bayesian networks [19]. The basic structure for knowledge representation in Bayesian logic programs are *Bayesian clauses* which model probabilistic dependencies between *Bayesian atoms* as in the following BLP corresponding to an example given in [4]:

$$c_1: \quad (likes(X,Y) \mid elephant(X),\ keeper(Y)) \qquad c_3: \quad likes(clyde, fred)$$
$$c_2: \quad (likes(X, fred) \mid elephant(X))$$

For each Bayesian clause c, a function cpd_c must be defined, expressing the conditional probability distribution $P(\mathsf{head}(c) \mid \mathsf{body}(c))$ and thus partially describing an underlying probability distribution P. For instance, $\mathsf{cpd}_{c_1}(\mathsf{true}, \mathsf{true}, \mathsf{true}) = 0.8$ would express our subjective belief that $likes(X, Y)$ is true with probability 0.8 if $elephant(X)$ and $keeper(Y)$ are true. In order to aggregate probabilities that arise from applications of different Bayesian clauses with the same head, BLPs make use of *combining rules*. Semantics are given to Bayesian logic programs via transformation into propositional forms, i. e. into Bayesian networks [25] (see [19] for details).

Markov Logic Networks. *Markov logic* [5] establishes a framework which combines Markov networks [25] with first-order logic to handle a broad area of statistical relational learning tasks. The Markov logic syntax complies with first-order logic where each formula is quantified by an additional weight value, e.g.

$$(elephant(X) \wedge keeper(Y) \Rightarrow likes(X,Y), \quad 2.2) \qquad (likes(clyde, fred),\ \infty)$$
$$(elephant(X) \Rightarrow likes(X, fred), \quad -0.8)$$

Semantics are given to sets of Markov logic formulas by a probability distribution over propositional possible worlds that is calculated as a log-linear model over weighted ground formulas. The fundamental idea in Markov logic is that first-order formulas are not handled as hard constraints (which are indicated by weight ∞), but each formula is more or less softened depending on its weight. A *Markov logic network (MLN)* L is a set of weighted first-order logic formulas $(F_i,\ w_i)$ together with a set of constants C. The semantics of L is given by a ground Markov network $M_{L,C}$ constructed from F_i and C [6]. The standard semantics of Markov networks [25] is used for reasoning, e.g. to determine the consequences of L (see [6] for details).

Relational Maximum Entropy. The syntax of *relational probabilistic conditional logic* (RPCL) [33] has already been used in the representation of the elephant-keeper-example from the introduction and employs conditionals of the form $(B \mid A)[x]$ with first-order formulas A, B and $x \in [0,1]$. A conditional $(B \mid A)[x]$ represents a constraint on a probability distribution $P : \Omega \to [0,1]$ on the set of possible worlds Ω and expresses that the conditional probability of B given A is x. In order to interpret conditionals containing free variables several relational semantics have been proposed, see [33,21]. The *grounding semantics* [21] uses a *grounding operator* $\mathcal{G}$, e. g. universal instantiation, that translates a set $\mathcal{R}$ of conditionals with free variables into a set of ground conditionals. Then, a probability distribution P *$\mathcal{G}$-satisfies* $\mathcal{R}$, denoted by $P \models_{\mathcal{G}} \mathcal{R}$, iff $P(B' \mid A') = x$ for every ground $(B' \mid A')[x] \in \mathcal{G}(\mathcal{R})$. Both *averaging* and *aggregating semantics* [18,33] do not require a grounding operator but interpret the intended probability x of a conditional with free variables only as a guideline for the probabilities of its instances, but the actual conditional probabilities of the instantiated formulas may differ from x. More precisely, for the averaging semantics, a probability distribution P *$\varnothing$-satisfies* $\mathcal{R}$, denoted by $P \models_{\varnothing} \mathcal{R}$, iff for every $(B \mid A)[x] \in \mathcal{R}$ it holds that $P(B_1 \mid A_1) + \ldots + P(B_n \mid A_n) = nx$ where $(B_1 \mid A_1), \ldots, (B_n \mid A_n)$ are the ground instances of $(B \mid A)$. In the aggregating semantics, a probability function P *$\odot$-satisfies* $\mathcal{R}$, denoted by $P \models_{\odot} \mathcal{R}$, iff for every $(B \mid A)[x] \in \mathcal{R}$ it holds that $P(B_1 \wedge A_1) + \ldots + P(B_n \wedge A_n) = x(P(A_1) + \ldots + P(A_n))$ where $(B_1 \mid A_1), \ldots, (B_n \mid A_n)$ are the ground instances of $(B \mid A)$. Note that all these three semantics are extensions of classical probabilistic semantics for propositional probabilistic conditional logic [15]. Based on any of these semantical notions the *principle of maximum entropy* ([23,15], see also Sec. 2.1), can be used for reasoning. By employing this principle one can determine the unique probability distribution that is the optimal model for a consistent knowledge base $\mathcal{R}$ in an information-theoretic sense via

$$P_{\mathcal{R}}^{ME_\circ} = \arg \max_{P \models_\circ \mathcal{R}} \mathcal{H}(P) \tag{3}$$

with $\circ$ being one of $\mathcal{G}$, $\varnothing$, or $\odot$. We abbreviate the approaches of reasoning based on the principle of maximum entropy with grounding, averaging, and aggregating semantics with $\mathrm{ME}_{\mathcal{G}}$, $\mathrm{ME}_{\varnothing}$, and $\mathrm{ME}_{\odot}$, respectively. We say that a formula $(B \mid A)[x]$ is $\circ$-inferred from $\mathcal{R}$ iff $P_{\mathcal{R}}^{ME_\circ} \models_\circ (B \mid A)[x]$ with $\circ$ being one of $\mathcal{G}$, $\varnothing$, or $\odot$.

3 Systems

There are various implementations of probabilistic logic. In the following, we give brief overviews on a selection of systems and toolboxes for probabilistic logics based on a propositional or a relational setting.

3.1 SPIRIT and Probabilistic Reasoning under Maximum Entropy

Reasoning in probabilistic conditional logic by employing the principle of maximum entropy [23,15] requires solving the numerical optimization problem $P^* =$

$\arg\max_{P \models \mathcal{R}} H(P)$ (cf. Sec 2.1). SPIRIT [29] is an expert system shell[2] implementing maximum entropy reasoning and solving this optimization problem. In order to tame the complexity of the optimization task which grows exponentially in the number of variables, SPIRIT generates a junction-tree of variable clusters, allowing to represent the global probability distribution by a product of marginal distributions. SPIRIT has been used successfully in various application domains, like medical diagnosis, project risk management, or credit scoring.

MEcoRe [8] is another system implementing reasoning for propositional probabilistic conditional logic under maximum entropy. While it does not employ a junction-tree modelling, but a straight-forward representation of the complete probability distribution, its focus is on flexibly supporting different basic knowledge and belief management functions like revising or updating probabilistic beliefs, or hypothetical reasoning in what-if mode.

3.2 Alchemy

Markov logic is implemented in the Alchemy system[3] [20]. Alchemy provides a wide range of functionalities for statistical relational learning and probabilistic logic inference. In particular, the consequences of a Markov logic network L defined via the ground Markov network $M_{L,C}$ (cf. Sec. 2.2) can be determined. With respect to learning, both weight learning as well as learning the structure of an MLN is supported. Applications of MLN realized with Alchemy include classifications tasks and social network modelling. In Sec. 4, we will report on some experiments using MLNs and Alchemy in medical diagnosis.

3.3 ProbCog

The *ProbCog* (Probabilistic Cognition for Cognitive Technical Systems) system[4] [13] is a software suite for statistical relational learning. ProbCog currently supports three knowledge representation approaches: Bayesian Logic Networks (BLNs), Adaptive Markov Logic Networks (AMLNs), and Markov Logic Networks (MLNs). For each approach, ProbCog provides several learning and inference algorithms, implemented either in Java or Python. ProbCog provides a sophisticated framework for relational data, which features, amongst others, a unified data model (which allows data conversion for all integrated approaches) and the generation of synthetic data (for learning experiments). The main focus of the ProbCog suite is on providing a comprehensive library of algorithms and powerful data structures for statistical relational learning, but it also includes some graphical interfaces for learning and querying, respectively.

3.4 KReator

KReator [32] is a toolbox for representing, learning, and automated reasoning with various approaches combining relational first-order logic with probabilities[5].

[2] http://www.fernuni-hagen.de/BWLOR/spirit/index.php
[3] http://alchemy.cs.washington.edu/
[4] http://ias.cs.tum.edu/research-areas/knowledge-processing/probcog
[5] http://kreator.cs.tu-dortmund.de/

The implementation of KReator is done in Java and mirrors its objective to support different approaches to relational probabilistic knowledge representation and reasoning. It strictly separates the internal logic and the user interface, employing an abstract command structure allowing easy modifications on both sides. In order to support the implementation of other approaches, KReator features a large library on first-order logic and basic probabilistic methods. Among others this library contains classes for formulæ, rules, conditionals, and various methods to operate on these. There is also a rudimentary implementation of Prolog available that can be used for specifying background knowledge as e. g. in BLPs. This integrated library is designed to support a fast implementation of specific approaches to statistical relational learning. The task of integrating a new approach into the KReator system is supported by a small set of interfaces that have to be implemented in order to be able to access the new approach from the user interface. There are interfaces for knowledge bases (which demands e. g. support for querying), file writers and parsers (for reading and writing the specific syntax of an approach), and learner. One thing to note is that both file writers and parsers have to work on strings only, all the cumbersome overhead of file operations and I/O is handled by KReator. With the help of a plugin-like architecture the developer of a new approach only has to be concerned with connecting her approach to KReator using these interfaces. Then all the benefits of an integrated development environment as provided by KReator are immediately accessible. Currently, KReator supports knowledge representation using BLPs, MLNs, the relational maximum entropy approach RME, as well as Relational Bayesian Networks [12], and Probabilistic Prolog [27]; support for Logical Bayesian Networks [7], Probabilistic Relational Language [10], and Relational Bayesian Networks [14] is in preparation.

Performing inference on MLNs is done using the Alchemy software package [20], a console-based tool for processing Markov logic networks. For BLPs, a reasoning component was implemented within KReator. To process ground $\mathrm{ME}_{\mathcal{G}}$ knowledge bases, KReator uses a so-called ME-adapter to communicate with a MaxEnt-reasoner. Currently, such adapters are supplied for the SPIRIT reasoner [29] and for MEcore [8] which are tools for processing (propositional) conditional probabilistic knowledge bases using maximum entropy methods.

ProbCog and KReator share some similarities with respect to their general approach to gather different knowledge representation approaches within one software framework, e. g. both systems feature some sort of unified data model for evidence or sample data. But the primary application focus of both systems differs significantly: ProbCog is developed for its intended practical application and integration in cognitive technical systems. So its primary focus is on providing a versatile and efficient framework for that specific purpose, therefore some sort of unified graphical user interface to the framework is not needed. In contrast, KReator's focus is on the typical workflow of a knowledge engineer, researcher, or developer. Therefore, KReator collects different approaches in an integrated graphical development environment (see Fig. 2 for a screenshot of

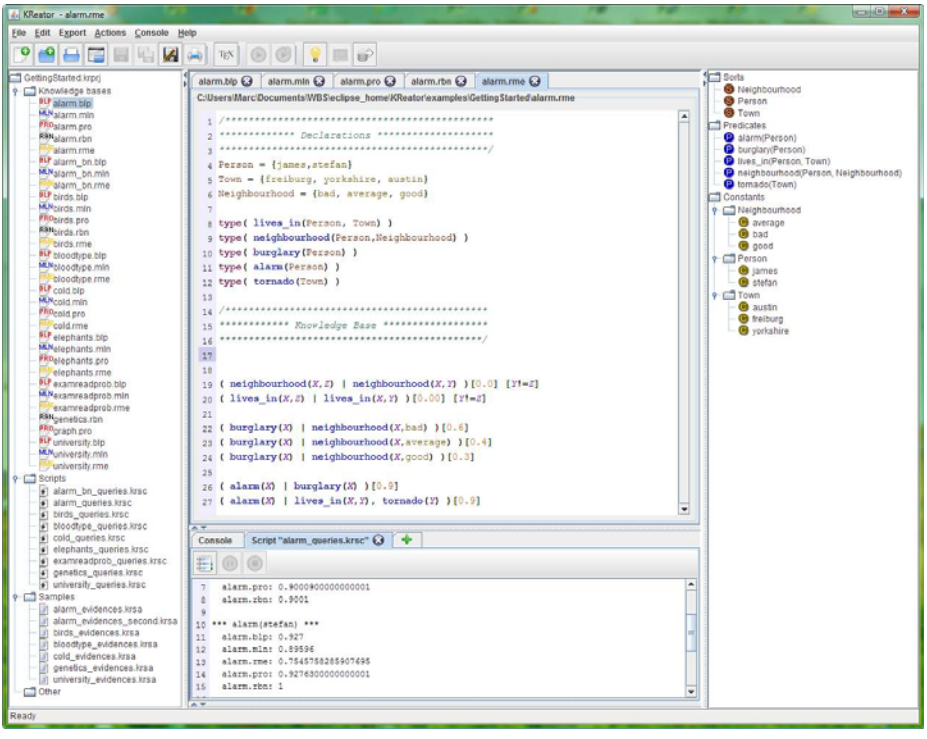


Fig. 2. Graphical user interface of the KREATOR system

the KREATOR user interface) to provide easy access to typical tasks and provides a plugin interface to support the study and development of further approaches.

4 Applications

In the following subsections, we will present three practical application scenarios of some of the afore described systems. All three applications cover settings from the medical domain. The first one illustrates ME-reasoning using a fictitious example, whereas the other ones describe learning experiments involving Markov logic networks and real-world data from medical studies.

4.1 Knowledge Processing with the MEcore System

In this section, we will illustrate how MECORE can process incomplete, uncertain knowledge expressed by a probabilistic knowledge base using a fictitious example from the medical domain. This example is taken from [8] and discusses the general treatment of a patient who suffers from a perilous bacterial infection. The infection will probably cause permanent neurological damage or even death if it is not treated appropriately. There are two antibiotics available that might

be capable of ending the infection, provided that the bacteria are not resistant to the specific antibiotic. It must also be considered that each antibiotic might cause a life-threatening allergic reaction that could be especially dangerous for an already weakened patient. The resistance of the bacteria to a specific antibiotic can be tested, but each test is very time-consuming.

Building Up the Knowledge Base. The construction of the knowledge base starts with the definition of some binary variables that describe aspects concerning antibiotic A:

`med_A`: The patient is treated with antibiotic A.
`effect_A`: Antibiotic A is effective against the bacteria.
`allergic_A`: The patient is allergic to antibiotic A.
`resistance_A`: The bacteria are resistant to antibiotic A.
`posResT_A`: The test result suggests a resistance to antibiotic A.

Analogously, there are also five variables concerning antibiotic B. A three-valued variable `outcome` describes the three possible outcomes of the treatment:

`outcome=healthy`: The infection is treated successfully and the patient is healthy again.
`outcome=impaired`: The patient overcomes the infection but suffers a permanent damage to the nervous system.
`outcome=dead`: The infection is not treated effectively and the patient dies.

The available knowledge summarizing the previously made experiences about the infection and the two antibiotics is modeled by the knowledge base `medKB` $= \{R_1, \ldots, R_{22}\}$ consisting of the probabilistic rules given in Fig. 3.

The first four rules express very obvious correlations between the variables: R_1 and R_2 say that if a certain antibiotic is not administered or the bacteria are resistent to it, then this antibiotic has no effect. R_3 and R_4 assure that if the bacteria are not resistant to a certain antibiotic, then this antibiotic is effective if—and only if—it is administered. The facts R_5 to R_9 integrate statistical information available for antibiotic A and antibiotic B, i. e. some a priori probabilities, into the knowledge base: antibiotic B is twice as likely as antibiotic A to cause an allergic reaction (R_5, R_6); and the resistance to antibiotic B is nine times higher compared to antibiotic A (R_7, R_8). It has occurred very rarely that somebody administers both antibiotics to the patient (R_9). R_{10} and R_{11} model the prognosis for the patient if no antibiotic is administered. The result of a resistance-test, testing the resistance of the bacteria to an antibiotic, always includes some error, but the test regarding antibiotic A is very reliable (R_{12}, R_{13}); whereas the test concerning antibiotic B has a somewhat lower sensitivity (R_{14}) and a considerably lower specificity (R_{15}).

The rules R_{16} to R_{19} express special knowledge about antibiotic A and antibiotic B, respectively: The allergic reaction caused by antibiotic A is most likely lethal (R_{16}), whereas the chance of surviving an allergy to antibiotic B is more likely than to die of it (R_{17}). If antibiotic A is effective, then the patient has a good chance to become healthy again (R_{18}), whereas the effectiveness of antibiotic B is somewhat lower (R_{19}). The following knowledge is available for

R_1 : (¬effect_A | ¬med_A ∨ resistance_A)[1.00]
R_2 : (¬effect_B | ¬med_B ∨ resistance_B)[1.00]
R_3 : (effect_A ⇔ med_A | ¬resistance_A)[1.00]
R_4 : (effect_B ⇔ med_B | ¬resistance_B)[1.00]
R_5 : (allergic_A)[0.10]
R_6 : (allergic_B)[0.20]
R_7 : (resistance_A)[0.01]
R_8 : (resistance_B)[0.09]
R_9 : (med_A ∧ med_B)[0.00001]
R_{10}: (outcome = dead | ¬med_A ∧ ¬med_B)[0.10]
R_{11}: (outcome = healthy | ¬med_A ∧ ¬med_B)[0.10]
R_{12}: (posResT_A | resistance_A)[0.97]
R_{13}: (¬posResT_A | ¬resistance_A)[0.99]
R_{14}: (posResT_B | resistance_B)[0.90]
R_{15}: (¬posResT_B | ¬resistance_B)[0.80]
R_{16}: (outcome = dead | med_A ∧ allergic_A)[0.99]
R_{17}: (outcome = dead | med_B ∧ allergic_B)[0.40]
R_{18}: (outcome = healthy | effect_A)[0.8]
R_{19}: (outcome = healthy | effect_B)[0.7]
R_{20}: (allergic_A | med_A)[0.10]
R_{21}: (outcome = dead | effect_B)[0.09]
R_{22}: (outcome = healthy | med_B ∧ allergic_B)[0.001]

Fig. 3. Probabilistic rules in the knowledge base medKB

antibiotic A only: R_{20} makes clear that the a priori probability of an allergy to antibiotic A (expressed by R_5 with equal probability) is not affected by the administration of antibiotic A. There is also some exclusive knowledge about antibiotic B: If antibiotic B is effective, there still remains some risk to die of the infection (R_{21}). If the patient survives an allergic reaction caused by antibiotic B, it is very unlikely that he will become healthy again (R_{22}).

Computing an Initial Epistemic State. In MEcore, the computation of an epistemic state incorporating the knowledge expressed by the knowledge base medKB can be initiated by the command:

(1) currState := epstate.initialize(medKB);

The calculated epistemic state currState represents the incomplete knowledge expressed by medKB inductively completed in an entropy-optimal way.

A closer look at medKB reveals that some additional rules can be logically deduced from the existing rules since they hold in all models satisfying medKB. For instance, a literal of the three-valued variable outcome makes up the conclusion of several rules. Hence, two rules with identical premise and an outcome literal as conclusion directly imply a corresponding third rule, e. g. R_{10} and R_{11} imply (outcome = impaired | ¬med_A ∧ ¬med_B)[0.8]. Appropriate queries to MEcore in currState yield these expected probabilities since reasoning at optimum entropy is compatible with classical probabilistic consequences.

Query. Suppose we want to know the patient's chances in each case of treatment, i. e. for each of the four possible options of medical administration: no antibiotic, antibiotic A only, antibiotic B only, both antibiotics. This can be expressed by a set of twelve query formulas (i. e. conditionals of the form e. g. $(\texttt{outcome}=\texttt{healthy} \mid \texttt{med\_A} \wedge \neg\texttt{med\_B})$) which we will denote by `medQueries`. While using classical probabilistic consequences does not yield informative answers for `medQueries`, MEcoRe infers the following probabilities from `currState`:

	healthy	impaired	dead
no antibiotic	0.10	0.80	0.10
only A	0.79	0.06	0.15
only B	0.65	0.23	0.12
A and B	0.94	0.02	0.04

These results clearly suggest that the combined administration of both antibiotics would be the best treatment. It offers a high chance of healing accompanied by a minimal risk of permanent neurological damage or death. However, a closer look at the knowledge base reveals that it implies that there is almost no possible drug interaction. For instance, asking for the degree of belief for the conditional

$$\texttt{C}_{\texttt{int}} : (\texttt{dead} \mid \texttt{med\_A} \wedge \texttt{med\_B} \wedge \neg\texttt{allergic\_A} \wedge \neg\texttt{allergic\_B})$$

in `currState` yields the inferred drug interaction probability 0.01.

Incorporation of New Knowledge. Suppose that later on, the doctors learn to know from an outside source that there is a severe risk (0.25) of a deadly drug interaction between both antibiotics. Executing

(2) $\texttt{currState.update}(\texttt{medKB}, \texttt{C}_{\texttt{int}}[0.25]);$

incorporates this new knowledge into the current epistemic state as if it had been available already in `medKB`. In fact, this kind of belief change is a *genuine revision* (cf. [17]) which in MEcoRe can also be more easily expressed by

(2') $\texttt{currState.revise}(\texttt{C}_{\texttt{int}}[0.25]);$

Now, asking the `medQueries` again, the probabilities have changed considerably (cf. Fig. 4(a)): With the knowledge about a deadly drug interaction, the probabilities show that the administration of antibiotic A maximizes the patient's chance to become healthy again.

What-If-Analysis. It has to be noticed that the knowledge used for generating the epistemic state `currState` says that no resistance tests have been performed, i. e. for neither of the antibiotics any resistance test results are available. A what-if-analysis can be used to analyze what changes would occur if a negative resistance-test result concerning antibiotic B was known. That is, could this test result make antibiotic B the better choice for treatment? In MEcoRe, such a what-if-analysis is accomplished by

(3) $\texttt{currState.whatif}((\neg\texttt{posResT\_B})[1.0], \texttt{medQueries});$

delivering the results shown in Fig. 4(b). The probabilities show that even a negative resistance-B test would not change the general decision to administer

(a)	healthy	impaired	dead
no antibiotic	0.10	0.80	0.10
only A	0.79	0.06	0.15
only B	0.65	0.23	0.12
A and B	0.70	0.02	0.28

(b)	healthy	impaired	dead
no antibiotic	0.10	0.80	0.10
only A	0.79	0.06	0.15
only B	0.69	0.21	0.10
A and B	0.76	0.02	0.22

(c)	healthy	impaired	dead
no antibiotic	0.10	0.80	0.10
only A	0.43	0.15	0.42
only B	0.65	0.23	0.12
A and B	0.32	0.05	0.63

(d)	healthy	impaired	dead
no antibiotic	0.10	0.80	0.10
only A	0.43	0.15	0.42
only B	0.54	0.26	0.20
A and B	0.20	0.04	0.76

Fig. 4. Probabilities for `medQueries` infered by MEcoRe

antibiotic A. This result is, amongst others, caused by the low resistance-B test specificity.

Another what-if-analysis revealing the effects of a positive resistance-A-test

(4) `currState.whatif((posResT_A)[1.0], medQueries);`

yields the probabilities given in Fig. 4(c). This shows that a test-result suggesting the resistance to antibiotic A would change the situation: In this case, a treatment with antibiotic B becomes the only that offers a realistic healing chance. This is not surprising, because a resistance-test result concerning antibiotic A is very reliable. So it is clearly advisable to perform the time-consuming resistance-A test.

In case of a positive resistance-A-test result, would it also be helpful to test the resistance to antibiotic B? That is, could an additional positive resistance-B-test change the decision to administer antibiotic B? Hypothetical reasoning

(5) `currState.whatif(((posResT_A)[1.0], (posResT_B)[1.0]), medQueries);`

yields the results shown in Fig. 4(d), indicating that even a positive resistance-B-test would not change the decision to administer antibiotic B. So it is not helpful to perform a resistance-B test in any situation, since its result would never change the decision that had been made without knowing the test result.

4.2 Diagnosis of Lung Cancer

This section is based on [9], reporting on a case study of using probabilistic relational modelling and learning as provided by MLNs and the MLN system Alchemy [20] in the field of biomedical diagnosis. The idea behind this diagnostical setting is to support diagnosis of bronchial carcinoma on the basis of the substances a person exhales [2,1]. In this setting, the focus is on the early detection of bronchial carcinoma by ion mobility spectrometry, a non-invasive diagnostic method which delivers results within a few minutes and can be applied at low costs.

Ion Mobility Spectrometry. In order to determine chemical substances in gaseous analytes, ion mobility spectrometry (IMS) can be used [2]. This method

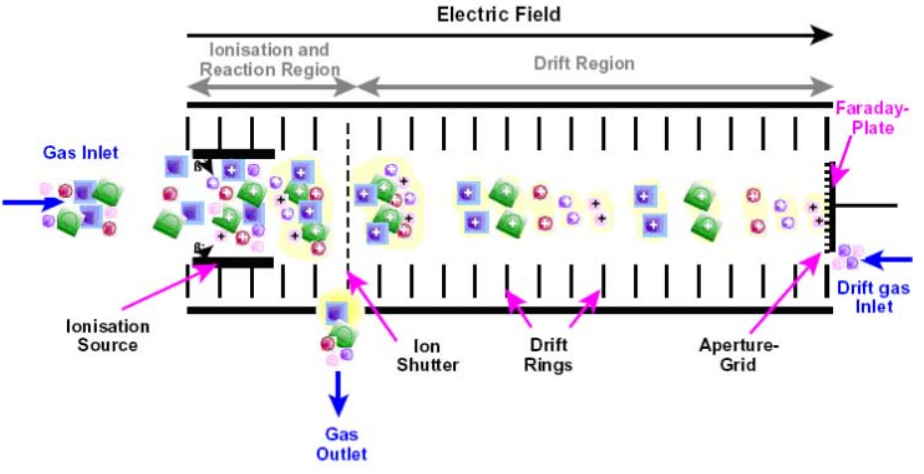


Fig. 5. Schematic overview of an ion mobility spectrometer (from [2])

relies on characterizing substances in gases by their ion mobility. Figure 5 illustrates the working principle of an ion mobility spectrometer. After ionisation, ion swarms enter the drift region through an ion shutter. The time needed to pass the drift region is called *drift time*, and the ion mobility is inversely proportional to the drift time. An ion mobility spectrum is obtained by mapping the drift time to the signal intensity measured at the Faraday plate (cf. Fig. 5). If the gaseous analyte contains various substances, they may reach the Faraday plate at the same time. In order to avoid this, a multi capillary column is used for the pre-separation of different substances [2] so that they enter the spectrometer at different time points, called *retention times*; for more detailed descriptions of ion mobility spectrometry and its working principle we refer to [1] or [2].

Thus, applying ion mobility spectrometry to gaseous analytes yields IMS spectra where a peak in such a spectrum corresponds to a particular substance. The determination of peaks in a measurement requires sophisticated processing of the raw spectra, see [2,3] for details. Peak objects taken from two different measurements that correspond to the same substance occur at corresponding areas in their respective so-called *heat maps*, and in order to identify such corresponding peaks, they can be mapped to *peak clusters* [2,9]. In our case study, we investigated an IMS database consisting of 158 measurements obtained from screening the breath of 158 patients out of which 82 had lung cancer (*bronchial carcinoma, bc*), yielding a database D_{bc} with 33 peak clusters, in the following referred to by the identifiers $pc0, \ldots, pc32$. For each peak cluster pc_i, $P(bc|pc_i)$ denotes the conditional probability that a measurement having a peak belonging to pc_i stems from a person having bronchial carcinoma. For applying methods of probabilistic relational modelling and learning to D_{bc}, we use a logic representation of D_{bc} (for convenience, also referred to as D_{bc}) involving the predicates $bc(M)$ indicating that measurement M belongs to a person having lung cancer and $pcInM(PC, M)$ stating that peak cluster PC occurs in measurement M.

In the following, we present different setups to learn MLNs from the data set D_{bc}. Our goal is to calculate the probability that a certain measurement m is from some person with a bronchial carcinoma, given the information for each of the 33 peak clusters whether or not it is contained in measurement m. That is, we want to calculate the conditional probability of $bc(m)$, given the truth values of the literals $pcInM(pc0, m)$, ..., $pcInM(pc32, m)$. This conditional probability helps to classify patients with respect to suffering from lung cancer. The corresponding classification task can be realized with MLNs. We use the software package Alchemy [20] which provides several sophisticated algorithms to perform (structure and parameter) learning and inference of MLNs. A learned MLN is validated in terms of classification accuracy, defined as the proportion of the correctly predicted (positive and negative) results on the total number of measurements in a testing set; these values are determined as the average accuracy of all tests in a 10-fold cross-validation.

Learning Logic Rules with the ILP System Aleph. In a first learning setup, we use the inductive logic programming (ILP) system *Aleph* [31] for learning first-order logic rules from the data set. Besides other parameters, Aleph allows to make detailed specifications about which atoms may appear in the body or head of a rule. As we want to predict whether or not the measurement M belongs to a patient having bronchial carcinoma, we require that heads of the rules learned by Aleph must contain the *bc* predicate, whereas their body must consist of one or more atoms of the *pcInM* predicate, with a constant in the first argument. This way, the rules predict the value of $bc(M)$, given the values of some of the $pcInM(pc_i, M)$. The two rules

$$R_1\colon\ pcInM(pc5, M) \wedge pcInM(pc8, M) \Rightarrow bc(M)$$
$$R_2\colon\ pcInM(pc7, M) \wedge pcInM(pc17, M) \wedge pcInM(pc31, M) \Rightarrow bc(M)$$

are examples of the 11 rules learned with Aleph [9]. The premises of all 11 rules consist of conjunctions of at most three positive *pcInM* literals. From the 33 different peak clusters found in the data set, only 18 occur in the rule set, so the other 15 peak clusters seem to carry no useful information with regard to lung cancer according to the Aleph result.

Learning Weights of Aleph Formulas with Alchemy. In a subsequent step, we take the Aleph implications as logical base structure of an MLN and learn appropriate weights for them from the data set using Alchemy. For instance, the resulting weights for the rules R_1 and R_2 above are 4.596 and 6.004, respectively. Evaluating the MLN prediction performance results in an accuracy of 78%.

If we take the implications as if-then-rules, we can determine the conditional probabilities of these rules under the distribution induced by the MLN, i. e. we use Alchemy to calculate the conditional probability of a rule's consequent ground atom given its premise ground atoms as evidence. E. g., for rule R_1, Alchemy determines the probability $P(bc(m)|pcInM(pc5, m) \wedge pcInM(pc8, m)) = 0.9800$ in the MLN; for R_2 we get 0.996. In fact, the conditional probabilities of all rules are not exactly 1.0, as expected, but rather close to it (see [9]). This is

due to the fact that Alchemy performs approximate inference and thereby, as a side-effect, prevents overfitting.

The learned MLN allows to draw some conclusions between peak clusters (i. e. the occurrence of substances in a measurement) and bronchial carcinoma. E. g., formula R_2 relates the combined occurrence of peak clusters *pc7*, *pc17*, and *pc31* in a measurement M to the presence of bronchial carcinoma. Because of the positive (and relative high) weight of this formula, the combined occurrence of these peak clusters can be interpreted as an indicator for bronchial carcinoma. Likewise, there are also formulas relating the combined occurrence of peak clusters to the absence of bronchial carcinoma.

Simple Classification with MLNs. In a further learning setup, we predefine the formula structure of a quite simple MLN: The MLN consists of the 33 implications $pcInM(pc0, M) \Rightarrow bc(M), \ldots, pcInM(pc32, M) \Rightarrow bc(M)$. Since the Alchemy syntax allows to express such "partially grounded" formulas in a compact way, the whole predefined structural Alchemy input merely consists of a single line. With this MLN structure, we follow a straightforwardly modelled classification approach: To classify the *bc* state of a measurement, we consider each peak cluster separately, leaving out any connections or dependencies among them. To some extent, this approach resembles Naive Bayes classification, where explicit independence assumptions among classifying attributes are made. The evaluation of the learned MLN revealed quite a high accuracy of 88% [9], although the enforced MLN structure lacks any connections between peak clusters, suggesting that those connections are not of great importance for classifying the measurements regarding *bc*.

MLN Structure Learning. In this learning setup, we make use of Alchemy's structure learning feature to learn an MLN from scratch. Alchemy does not allow to make detailed specifications about the formulas to be learned, i. e. we cannot impose the requirement that the $pcInM(\_, \_)$ atoms have a constant in the first argument. As a consequence, Alchemy's structure learning algorithm produces no useful results when applied to D_{bc} without any further information. So we modify the relational modelling in some aspect by replacing the binary predicate $pcInM(PC, M)$ by 33 unary predicates $pc0(M), \ldots, pc32(M)$.

Using this setup, the structure (and weight) learning with Alchemy starts from an empty MLN and results in an MLN with 89 formulas (including 34 atomic formulas for all 34 predicates) [9]. The evaluation of this MLN shows an accuracy of 90%. Compared to the previous results, this MLN models much more connections among the peak clusters and their combined influence regarding $bc(M)$. Only 13 of the 55 non-atomic formulas involve a *bc* literal, so the other 42 formulas express connections among the peak clusters regardless of the $bc(M)$ state, and the formulas contain both positive and negative peak cluster literals. So compared to the previous results, this MLN exhibits more complex and subtle connections among the occurrences of peak clusters and the $bc(M)$ state. Here are two examples for the learned formulas:

$$R_{61}: \quad (\neg pc10(M) \wedge pc14(M) \wedge \neg pc18(M) \wedge pc21(M) \Rightarrow \quad bc(M), \quad 7.15)$$
$$R_{44}: \quad (pc17(M) \wedge pc28(M) \Rightarrow pc21(M), \quad 5.05)$$

R_{61} relates the combined occurrence of peak clusters *pc14* and *pc21* and the explicit absence of peak clusters *pc10* and *pc18* in a measurement to bronchial carcinoma. With a lower, but still relatively high weight, R_{44} implies that a measurement containing peak clusters *pc17* and *pc28* also contains peak cluster *pc21*. In other words, the system has learned the relationship that the occurrence of the two substances indicated by peak clusters *pc17* and *pc28* in a measurement M leads to the presence of the substance identified by *pc21* in the same measurement. Such a relation can provide interesting insights into the general composition of substances in typical measurements.

4.3 Predicting Allergic Diseases of Children

In this section, another application of MLNs for modelling and learning in the medical domain is presented. In [30], MLNs were employed to analyze the correlations between allergic diseases of children and certain environmental factors. The data used in this analysis has been extracted from the KiGGS study of the Robert Koch-Institut [28]. The KiGGS study is a long term study which covers the health situation of 17.000 children (and adolescents) in Germany. It considers a multitude of attributes for every child concerning medical or social aspects. For the experiments described in [30], 13 of these attributes had been chosen which represent well-known risk factors for allergies, e. g. *"the child has a pet at home"*, *"the child lives in an urban environment"*, or *"a parent suffers from an allergy"*. Each such attribute was modelled by a corresponding MLN predicate, e. g. $hasPet(X)$, $urban(X)$. Together with the information whether or not a child is allergic (represented by an $isAllergic(X)$ predicate) this allowed to model the data from the study as MLN learning data, i. e. as data samples in terms of ground atoms. The extracted and preprocessed learning data from the study consisted of about 8.000 data samples, covering allergic respectively non-allergic children in equal parts. In all experiments, subsets of these data samples were used as actual training and testing data (performing a 5-fold cross-validation).

Several learning experiments were performed on this learning data using the algorithms of the Alchemy software package [20] for learning and inference. The goal of all experiments was to learn an MLN which can predict the risk of a child to be allergic given the presence (or absence, respectively) of each of the 13 risk factors. The learning experiments included parameter (i. e. weight) learning using a predefined MLN formula structure which consisted of 13 implications of the form e. g. $hasPet(X) \Rightarrow isAllergic(X)$. In another experiment, Alchemy's structure learning algorithm was applied to learn an MLN (formulas and weights) from scratch. The evaluation of the learned MLNs was carried out by using several of Alchemy's (approximate) inference algorithms. Additionally, the software PyMLNs (which is part of the ProbCog suite [13]) was used to perform exact inference on some MLNs in order to evaluate the deviation compared to the approximate results. The experiments showed that the results of the

various Alchemy algorithms were quite similar and that there were no significant difference compared to the exact results.

Overall, the quality of the learned MLNs in terms of classification accuracy turned out to be not as good a expected. For various experiment settings, the MLNs resulting from structure as well as from parameter learning provide an accuracy of about 61% in predicting a child to be allergic. This could be improved by focusing on formulas the probabilities of which were significantly different from 0.5. However, further investigations into the evaluation of the quality of learned MLNs for prediction tasks in this domain will be necessary.

5 Summary and Conclusion

This paper gives a brief overview on the state of the art in probabilistic reasoning, and illustrates the relevance of probabilistic methods for expert systems by describing their applications in various scenarios. The main advantage of probabilistic formalisms is an accurate handling of uncertainty which pervades all real world problems. Degrees of uncertainty can be conveniently obtained from statistical data and processed via probabilistic networks. Moreover, we go into more details on novel approaches combining probability theory and first-order logic which provide more expressive frameworks for probabilistic reasoning. The problem of incompleteness of knowledge is addressed by describing the information-theoretical principle of maximum entropy which might also be applied in first-order settings. Altogether, probabilistic frameworks provide suitable and rich environments for learning, modelling, and reasoning in expert systems.

References

1. Baumbach, J., Bunkowski, A., Lange, S., Oberwahrenbrock, T., Kleinbölting, N., Rahmen, S., Baumbach, J.I.: IMS^2 – An integrated medical software system for early lung cancer detection using ion mobility spectometry data of human breath. J. of Integrative Bioinformatics 4(3) (2007)
2. Baumbach, J.I., Westhoff, M.: Ion mobility spectometry to detect lung cancer and airway infections. Spectroscopy Europe 18(6), 22–27 (2006)
3. Bödeker, B., Vautz, W., Baumbach, J.I.: Peak finding and referencing in MCC/IMS-data. International Journal for Ion Mobility Spectrometry 11(1-4), 83–87 (2008)
4. Delgrande, J.P.: On First-Order Conditional Logics. Artificial Intelligence 105(1-2), 105–137 (1998)
5. Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for Artificial Intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool, San Rafael (2009)
6. Domingos, P., Richardson, M.: Markov logic: A unifying framework for statistical relational learning. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)
7. Fierens, D., Blockeel, H., Ramon, J., Bruynooghe, M.: Logical Bayesian Networks. In: Dzeroski, S., Blockeel, H. (eds.) Proceedings of the 3rd International Workshop on Multi-Relational Data Mining, pp. 19–30 (2004)

8. Finthammer, M., Beierle, C., Berger, B., Kern-Isberner, G.: Probabilistic reasoning at optimum entropy with the MEcore system. In: Lane, H.C., Guesgen, H.W. (eds.) Proceedings 22nd International FLAIRS Conference, FLAIRS 2009. AAAI Press, Menlo Park (2009)
9. Finthammer, M., Beierle, C., Fisseler, J., Kern-Isberner, G., Baumbach, J.I.: Using probabilistic relational learning to support bronchial carcinoma diagnosis based on ion mobility spectrometry. International Journal for Ion Mobility Spectrometry 13, 83–93 (2010)
10. Getoor, L., Grant, J.: Prl: A probabilistic relational language. Machine Learning 62(1), 7–31 (2006)
11. Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)
12. Jaeger, M.: Relational Bayesian Networks: A Survey. Electronic Transactions in Artificial Intelligence 6 (2002)
13. Jain, D., Mösenlechner, L., Beetz, M.: Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities. In: International Conference on Robotics and Automation (ICRA), pp. 3130–3135 (2009)
14. Jain, D., Waldherr, S., Beetz, M.: Bayesian Logic Networks. Tech. rep., IAS Group, Fakultät für Informatik, Technische Universität München (2009)
15. Kern-Isberner, G.: Characterizing the principle of minimum cross-entropy within a conditional-logical framework. Artificial Intelligence 98, 169–208 (1998)
16. Kern-Isberner, G.: Conditionals in Nonmonotonic Reasoning and Belief Revision. LNCS (LNAI), vol. 2087. Springer, Heidelberg (2001)
17. Kern-Isberner, G.: Linking iterated belief change operations to nonmonotonic reasoning. In: Brewka, G., Lang, J. (eds.) Proceedings 11th International Conference on Knowledge Representation and Reasoning, KR 2008, pp. 166–176. AAAI Press, Menlo Park (2008)
18. Kern-Isberner, G., Thimm, M.: Novel Semantical Approaches to Relational Probabilistic Conditionals. In: Proc. Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010), pp. 382–392 (2010)
19. Kersting, K., De Raedt, L.: Bayesian logic programming: Theory and tool. In: Getoor, L., Taskar, B. (eds.) Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)
20. Kok, S., Singla, P., Richardson, M., Domingos, P., Sumner, M., Poon, H., Lowd, D., Wang, J.: The Alchemy System for Statistical Relational AI: User Manual. Department of Computer Science and Engineering. University of Washington (2008)
21. Loh, S., Thimm, M., Kern-Isberner, G.: On the problem of grounding a relational probabilistic conditional knowledge base. In: Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR 2010), Toronto, Canada (May 2010)
22. Nute, D., Cross, C.: Conditional Logic. In: Gabbay, D., Guenther, F. (eds.) Handbook of Philosophical Logic, vol. 4, pp. 1–98. Kluwer Academic Publishers, Dordrecht (2002)
23. Paris, J.: The uncertain reasoner's companion – A mathematical perspective. Cambridge University Press, Cambridge (1994)
24. Paris, J.: Common sense and maximum entropy. Synthese 117, 75–93 (1999)
25. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1998)
26. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1998)

27. Raedt, L.D., Kimmig, A., Gutmann, B., Kersting, K., Costa, V.S., Toivonen, H.: Probabilistic Inductive Querying Using ProbLog. Tech. Rep. CW 552, Department of Computer Science, Katholieke Universiteit Leuven, Belgium (June 2009)
28. Robert Koch-Institut: Public Use File KiGGS, Kinder- und Jugendgesundheitssurvey 2003-2006, Berlin (2008)
29. Rödder, W., Reucher, E., Kulmann, F.: Features of the expert-system-shell SPIRIT. Logic Journal of the IGPL 14(3), 483–500 (2006)
30. Schmaußer-Hechfellner, E.: Probabilistische logikbasierte Wissensmodellierung mit statistischen medizinischen Daten unter Verwendung von Lern- und Inferenzverfahren für Markov-Logik-Netze. Bachelor Thesis, Dept. of Computer Science, FernUniversität in Hagen (2011) (in German)
31. Srinivasan, A.: The Aleph Manual (2007), http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/
32. Thimm, M., Finthammer, M., Loh, S., Kern-Isberner, G., Beierle, C.: A system for relational probabilistic reasoning on maximum entropy. In: Guesgen, H.W., Murray, R.C. (eds.) Proceedings 23rd International FLAIRS Conference, FLAIRS 2010, pp. 116–121. AAAI Press, Menlo Park (2010)
33. Thimm, M., Kern-Isberner, G., Fisseler, J.: Relational probabilistic conditional reasoning at maximum entroy. In: Proceedings ECSQARU-2011 (to appear, 2011)

Multi-objective Optimal Combination Queries

Xi Guo and Yoshiharu Ishikawa

Graduate School of Information Science,
Nagoya University,
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
guoxi@db.itc.nagoya-u.ac.jp, ishikawa@itc.nagoya-u.ac.jp
http://www.db.itc.nagoya-u.ac.jp/en/

Abstract. Multi-objective optimization problem finds out optimal objects w.r.t. several objectives rather than a single objective. We propose a new problem called a *multi-objective optimal combination problem* (*MOC problem*) which finds out object combinations w.r.t. multiple objectives. A combination dominates another combination if it is not worse than anther one in all attributes and better than another one in one attribute at least. The combinations, which cannot be dominated by any other combinations, are optimal. We propose an efficient algorithm to find out optimal combinations by reducing the search space with a lower bound reduction method and an upper bound reduction method based on the R-tree index. We implemented the proposed algorithm and conducted experiments on synthetic data sets.

Keywords: Multi-objective, combination, R-tree, domination.

1 Introduction

The *multi-objective optimization problem* [1,2] finds out objects which are optimal w.r.t. several objectives rather than a single objective. In this paper, we propose a new variation which finds out optimal object combinations w.r.t. multiple objectives. We name it a *multi-objective optimal combination* (*MOC*) problem. Let us consider an example first.

Example 1. *A user wants to buy a breakfast consisting of three foods. Her budget is 1300JPY and her calorie demand is 1600kcal. Assume that six different foods are available in Fig. 1 (a)*[1]*. All 3-item food combinations are shown in Fig. 1 (b) with (cost, calorie) and are also shown in Fig. 1 (c) as points. We need to recommend better combinations for her.*

The combinations $\{AED\}$, $\{ACB\}$, $\{AEB\}$, $\{ACE\}$, $\{CED\}$, $\{AEF\}$ *and* $\{CEB\}$ *are within the user's requirements* $(13, 16)$ *as Fig. 1 (c) shows. They are possible answers for the user. The combination* $\{CEB\} = (13, 14)$ *is better than the combination* $\{ACE\} = (9, 12)$ *because* $\{CEB\}$ *is closer to the requirements* $(13, 16)$*. We say that* $\{CEB\}$ *dominates* $\{ACE\}$*.*

[1] The cost unit is 100 JPY and the calorie unit is 100kcal.

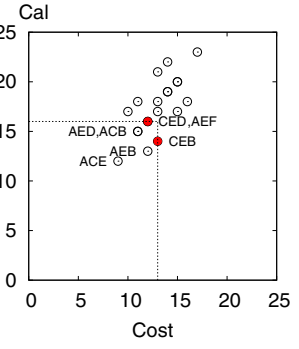
A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 47–61, 2011.

Food	(Cost, Cal)
A	(2,4)
B	(6,6)
C	(3,5)
D	(5,8)
E	(4,3)
F	(6,9)

(a) Food

Comb'n	(Cost, Cal)	Comb'n	(Cost, Cal)
ACB	(11,15)	ABF	(14,19)
ACD	(10,17)	ACF	(11,18)
AEB	(12,13)	ADF	(13,21)
AED	(11,15)	AEF	(12,16)
CEB	(13,14)	BCF	(15,20)
CED	(12,16)	BDF	(17,23)
ABD	(13,18)	BEF	(16,18)
CBD	(14,19)	CDF	(14,22)
EBD	(15,17)	CEF	(13,17)
ACE	(9,12)	DEF	(15,20)

(b) Three-Foods Combination



(c) Optimal Food

Fig. 1. MOC Problem Example

Suppose there combinations, which cannot be dominated by any other combinations, are optimal ones to be recommended. In this example, $\{CED\}$, $\{AEF\}$ *and* $\{CEB\}$ *(solid points) cannot be dominated. We return them to the user as the results. The challenge of the problem is that there will be a huge number of combinations consisting of elements from a given object set and we need to identify the optimal combinations. This example is going to act as a running example in the rest of this paper.* ■

There is an object set G where each object has m attributes $(g^1, g^2, \cdots, g^m)$. An h-item combination $p = \{g_1, g_2, \cdots, g_h\}(g_i \in G)$ has attributes $(p^1, p^2, \cdots, p^m)$ where $p^j = \Sigma_{i=1}^{h} g_i^j$ $(j \in 1, 2, \cdots, m)$. Given an objective vector $\boldsymbol{b} = (b^1, b^2, \cdots, b^m)$, the *distance* from a combination p to $\boldsymbol{b}$ is $(d^1, \cdots, d^m)$ where $d^j = b^j - \Sigma_{i=1}^{h} g_i^j$. If $d^j \geq 0$ for all j, the combination p is *eligible* to be an optimal combination.

Definition 1 (Domination). *Given an objective vector* $\boldsymbol{b}$*, one eligible combination* $\boldsymbol{p}$ dominates *another eligible combination* $\boldsymbol{p'}$ *if* $d^k < d'^k$ $(k \in 1..m)$ *and* $d^j \leq d'^j$ $(j \in 1..m$ and $j \neq k)$. □

Definition 2 (Multi-Objective Optimal Combination). *If an* h*-item combination cannot be dominated by any other combinations* $p_i \in P - \{p\}$*, it is a multi-objective optimal combination (MOC).* □

Problem 1 (MOC Query). *Given an object set* G*, an objective vector* $\boldsymbol{b}$ *and a combination cardinality* h*, an* MOC query *finds out the MOC set* $S = \{s_1, s_2, \cdots, s_l\}$ *where* s_i $(i \in 1, 2, \cdots, l)$ *is an optimal combination consisting of* h *objects.* □

A naïve method to solve the MOC problem is to enumerate all possible h-item combinations and decide whether they are dominated or not. The non-dominated

ones are returned as optimal combinations. However, this method is very time-consuming. In this paper, we propose an efficient algorithm to find out optimal ones using a lower bound and an upper bound reduction methods. The two reduction methods are based on the R-tree which indexes objects using hierarchical minimum bounding rectangles (MBRs) [11]. We construct combinations by searching an R-tree in a depth-first way. Considering lower bounds and upper bounds of MBRs, we reduce the search space and obtain candidates quickly. Finally, we find out the optimal ones from the candidates.

We first review some studies related to the proposed MOC problem in Section 2. Next, we propose the algorithm to answer MOC queries in Section 3. In Section 4, we report experimental results and conclude the paper in Section 5.

2 Related Work

In databases area, multi-objective optimization problems have received considerable attentions since the first work [2] proposed a skyline query problem. The skyline query problem aims at finding out optimal objects which cannot be dominated by any other objects. One object dominates another object if it is not worse than another one in all attributes and better than another one in one attribute at least. Many subsequent algorithms are proposed to improve the performances of skyline queries, like BBS [8], SFS [12] and LESS [13]. Our MOC query problem, however, is different from the classical skyline query problem because it focuses on object combinations rather than objects themselves. Though an object combination can be regarded as an object with aggregation attribute values of its elements, it is time consuming to use an existing algorithm to solve the MOC problem because there will be a huge number of object combinations to be processed.

The research of skyline queries on object combinations is limited. To the best of our knowledge, the first and only work on this topic is "top-k combinatorial skyline queries" [3]. This research was motivated by the investment portfolio which finds out optimal stock combinations considering profit and risk attributes. The authors studied how to find out top-k non-dominated combinations which rank from 1 to k before other non-dominated ones according to a given preference order in attributes. They constructed non-dominated combinations incrementally considering the preference order and terminates as soon as the top-k results have been found. However, our MOC query problem simply focuses on finding out non-dominated combinations rather than a top-k query with some preference orders.

One may think that our MOC query problem seems alike to the zero-one knapsack problem [14] which is in the linear integer programming category [6]. Given each object has a value attribute and a weight attribute. A knapsack problem finds out the best object combination with a maximum total value and within a total weight limitation. The knapsack problem aims at optimizing the value attribute within a weight constraint. However, our MOC problem is to find out trade-offs between the value attribute and the weight attribute.

In order to solve our MOC query problem, we organize objects using the R-tree index [11] and retrieve object combinations using a lower bound reduction method and an upper bound reduction method. Our lower bound reduction method employs the basic idea of the forward checking (FC) algorithm [7] which constructs combinations incrementally to answer structural queries in spatial databases. A structural query asks for object combinations which have a spatial structure similar to a required structure. Our upper bound reduction method employs the basic idea of the BBS algorithm [8] which is an efficient solution for classical skyline queries [2] on objects rather than on object combinations in our MOC query problem.

3 Algorithms

Given objects indexed by an R-tree, we construct MBR combinations in a depth-first way until reaching the leaf level where the MBRs are real objects. Each MBR combination can be expanded using its child MBRs. Let us use Example 2 to illustrate how to retrieve combinations using the R-tree index.

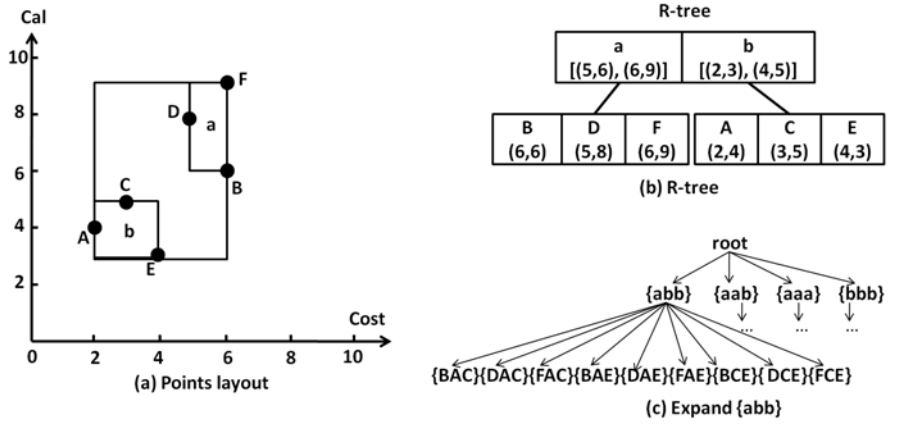


Fig. 2. Construct combinations using R-tree

Example 2. *Fig. 2 (a) and Fig. 2 (b) show the R-tree index of objects in the running example. Let us construct 3-item combinations using the R-tree. There are two MBRs a and b at the root level. We can select one object from MBR a and select two objects from MBR b to construct a 3-item combination. For simple, we use MBR combination* $\{abb\}$ *to denote this selection pattern. As Fig. 2 (c) shows, we can expand pattern* $\{abb\}$ *using objects involved in MBR a and objects involved in MBR b. Nine combinations (i.e.,* $\{BAC\}$ *to* $\{FCE\}$*) are obtained following pattern* $\{abb\}$*. In the same way, we can generate object combinations following patterns* $\{aab\}$*,* $\{aaa\}$ *and* $\{bbb\}$*.* ■

Example 2 illustrates that we can construct h-item combinations easily by retrieving the R-tree in a depth-first way. The depth-first retrieval provides us

an opportunity to reduce the search space by eliminating non-promising MBR combinations (i.e., patterns). If we can eliminate non-promising MBR combinations before they are expanded to real object combinations, we need fewer comparisons for object combinations at the leaf level. A lower bound reduction method and an upper bound reduction method are proposed to eliminate the non-promising MBR combinations.

3.1 Lower Bound Reduction

An MBR combination has a lower bound which is an aggregation on the lower bounds of its elements. For example, in Figure 2 the combination $\{abb\}$ has a lower bound $\{abb\}^{\perp} = (9, 12)$ which is an aggregation on the lower bounds of its elements one a and two b, namely, $\{abb\}^{\perp} = a^{\perp} + b^{\perp} \times 2 = (5, 6) + (2, 3) \times 2$. We define it formally as follows.

Definition 3 (Lower Bound for MBR Combination). *An MBR combination* $p = \{e_1, e_2, \cdots, e_h\}$ *has a* lower bound $p^{\perp}$ *which is an aggregation on the lower bounds of its elements* e_1 *to* e_h*, namely,* $p^{\perp} = \Sigma_{i=1}^{h} e_i^{\perp}$ *where* $e_i^{\perp}$ *is the lower bound of* e_i*.* □

Theorem 1. *Given an objective vector* $\boldsymbol{b} = (b^1, b^2, \cdots, b^m)$*, an MBR combination* p *cannot be expanded to optimal object combinations, if its lower bound* $p^{\perp}$ *is beyond of the objective vector* $\boldsymbol{b}$*, namely,* $p^{i\perp} > b^i$ $(i \in 1, 2, \cdots, m)$*.* □

Proof 1. *We expand an MBR combination* $p = \{e_1, e_2, \cdots, e_h\}$ *using child MBRs of* e_1 *to* e_h *until we reach the leaf level. In other words, we select objects enclosed in* e_i $(i \in 1, 2, \cdots, h)$ *to construct object combinations. Every object* g_i *selected from* e_i *has attribute values* $g_i^j \geq e_i^{j\perp}$ $(j \in 1, 2, \cdots, m)$*. An object combination consisting of these objects has attribute values* $\Sigma_{i=1}^{h} g_i^j \geq p^{j\perp}$ *where* $p^{j\perp} = \Sigma_{i=1}^{h} e_i^{j\perp}$*. If* $p^{j\perp} > b^j$*, the combination is not eligible to be an optimal one because its attribute value* $\Sigma_{i=1}^{h} g_i^j > b^j$*.* ■

Example 3. *Let us think about constructing 3-item combinations again. Fig. 3 (a) shows the lower bounds of MBR combinations. Given an objective vector* $(13, 16)$*, we prune the MBR combination* $\{aaa\}$ *because it has a lower bound* $[15, 18]$ *which is beyond of* $(13, 16)$*. Combination* $\{aaa\}$ *will not be expanded.* ■

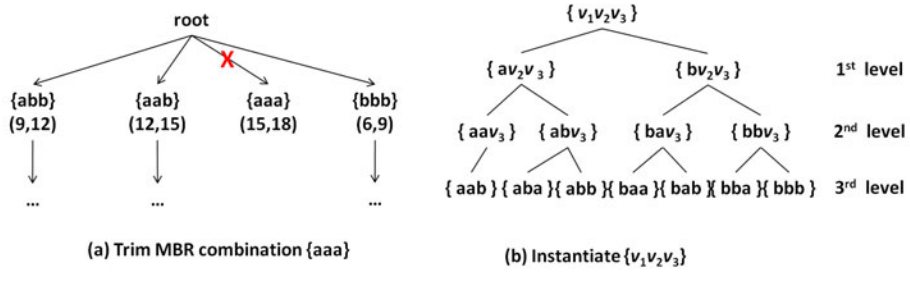


Fig. 3. Lower Bound Reduction

In order to generate MBR combinations with lower bounds within the objectives, we incrementally obtain them using a method inspired by the *forward checking* algorithm used in [7]. An h-item combination can be denoted as $v_1 v_2 \cdots v_h$ ($v_1 \in C_1,\ v_2 \in C_2, \cdots,\ v_h \in C_h$) where C_i is the domain of v_i. We instantiate variable v_i by selecting an MBR or an object from domain C_i. Our instantiation order is from v_1 to v_h. We obtain a combination $c_1 c_2 \cdots c_h$ after instantiating v_h at last.

During the process, after instantiating an intermediate variable v_{l-1}, we obtain a partial combination $c_1 c_2 \cdots c_{l-1} v_l \cdots v_h$ where $c_i \in C_i$ and $i \in [1, l-1]$. The process is at the l^{th} *instantiation level* where v_{l-1} has been instantiated while v_l needs to be instantiated. The domain C_l for v_l is decided by the current partial combination $c_1 c_2 \cdots c_{l-1} v_l \cdots v_h$. The MBRs in domain C_l should have lower bounds within $T = \boldsymbol{b} - \Sigma_{i=1}^{l-1} c_i^{\perp}$.

Example 4. *Fig. 3 (b) shows the process of instantiating a combination $v_1 v_2 v_3$. Let us take the leftmost branch as an example. At the 1^{st} level, we instantiate v_1. Given the objective vector $(13, 16)$, domain C_1 is $\{a, b\}$. After setting MBR a to variable v_1, we obtain a partial combination $\{a v_2 v_3\}$. Next, at the 2^{nd} level, we instantiate v_2 and objects belongs to C_2 should have lower bounds within $(8, 10) = (13, 16) - (5, 6)$ where $a^{\perp} = (5, 6)$. Domain C_2 is $\{a, b\}$ and we set MBR a to variable v_2. Now the partial combination is $\{a a v_3\}$. Next, at the 3^{rd} level, we instantiate v_3 and objects belongs to C_3 should have lower bounds within $(3, 4) = (8, 10) - (5, 6)$. Domain C_3 is $\{b\}$ and we set MBR b to variable v_3. Finally, we obtain a combination $\{aab\}$. In the same way, we can obtain other combinations.*

Notice that there are duplicate combinations generated during the lower bound reduction process. Two combinations are duplicates if they have same elements regardless of their element orders (e.g. $\{aab\}$ and $\{baa\}$). It is easy to remove such duplicates and we will not talk it too much for the space limitation. ■

Algorithm 1 shows the process of MOC queries using the lower bound reduction method. We start a query process by calling a function MOC_query$(p, \boldsymbol{b}, h, S)$ where $p = \{root,\ root,\ root\}$ and $S = \emptyset$. We use d_{ji} to denote the domain C_i for variable v_i at the j^{th} instantiation level. We first initialize the threshold T as $\boldsymbol{b}$, initialize d_{1i} ($i \in 1, 2, \cdots, h$) as child MBRs of e_i using a function get_children(e_i), and initialize the current instantiation level identifier l as 1 (from line 3 to 6). Next, we expand the combination p (line 7 to line 30).

From line 9 to 18, we instantiate the variable v_l. We select an MBR from d_{ll} to instantiate v_l using a function get_MBR(d_{ll}) (line 10). At the same time, the function get_MBR(d_{ll}) removes the selected MBR from d_{ll}. If d_{ll} is empty, we backtrack to the level $(l-1)$ (line 12 to 18). Note that we will not do the backtrack operation if the current level is 1 (line 12 to 13).

From line 19 to 24, we prepare domains for the next instantiation level $(l+1)$ using the function forward_check(). After updating the threshold T considering the instantiated variables (line 21), we call a function forward_check(T, l, i) (line 31 to 38). In the function, we initialize domains $d_{l+1,j}$ ($j \in i+1, i+2, \cdots, h$) as domains $d_{l,j}$ at the previous level l. We check each MBR in $d_{l+1,j}$ and remove the ones which have lower bounds beyond T (line 35 to 37).

Algorithm 1. MOC Query Using Lower Bound Reduction

```
1: procedure MOC_query(p, b, h, S)   {p = e1e2···eh is a combination to be
   expanded; S contains optimal object combinations.}
2: p' := v1v2···vh;   {Expand p to p' which have h variables to instantiate.}
3: T := b;   {Initialize threshold T as b.}
4: for i := 1 to h do
5:     d1i := get_children(ei);   {Initialize domains d1i.}
6: l := 1;   {Start from the 1st instantiation level.}
7: while true do
8:     begin
9:     if dll ≠ ∅ then   {MBRs in dll are not used up.}
10:        vl := get_MBR(dll);   {Select an MBR from dll to instantiate vl.}
11:    else   {MBRs in dll are used up.}
12:        if l = 1 then
13:            return;   {Terminate the expansion of p.}
14:        else
15:            begin
16:            l := l − 1;
17:            continue;   {Backtrack to level (l − 1).}
18:            end
19:    if l < h then   {At a level before the last level h.}
20:        begin
21:        T := T − v_l^⊥;   {Update T.}
22:        forward_check(T, l, i);   {Prepare domains for level (l + 1).}
23:        l := l + 1;   {Start the instantiation for level (l + 1)}
24:        end
25:    else   {At the last level h.}
26:        if at_leaf_level(p) then
27:            update_optimal_set(p', S);   {Update S considering p'.}
28:        else
29:            MOC_query(p', b, h, S);   {Expand p'.}
30:    end
31: procedure forward_check(T, l, i)
32: for j := i + 1 to h do
33:    begin
34:    dl+1,j = dl,j;   {Initialize domains at level l + 1.}
35:    for k := 1 to n do   {dl+1,j = {ck|k ∈ 1, 2, ···, n}.}
36:        if is_beyond(c_k^⊥, T) then   {c_k^⊥ is beyond T.}
37:            dl+1,j := dl+1,j − {ck};   {Eliminate ck from dl+1,j.}
38:    end
```

Let us go back to the function MOC_query(). If we are not expanding a combination at the leaf level, we recursively call the function MOC_query() to expand a newly generated combination p' (line 29). If not, we update the optimal object combination set S (line 27). A function update_optimal_set(p', S) decides whether a new object combination p' can be dominated by an existing combination in S. We add it into S, if it cannot be dominated by any combinations in S. The combinations in S, which is dominated by p', are removed.

3.2 Upper Bound Reduction

We obtain optimal object combinations and inserting them into the set S while retrieving the R-tree in a depth first way as Algorithm 1 shows. An MBR combination is promising if it has an upper bound which cannot be dominated by any combinations in S. This *upper bound reduction* method avoids expanding MBR combinations which will generate combinations dominated by others.

Definition 4 (Upper Bound for MBR Combination). *An MBR combination* $p = \{e_1, e_2, \cdots, e_h\}$ *has an* upper bound $p^\top$ *which is an aggregation on the upper bounds of its elements* e_1 *to* e_h*, namely,* $p^\top = \Sigma_{i=1}^{h} e_i^\top$ *where* $e_i^\top$ *is the upper bound of* e_i*.* □

Definition 5. *Given an objective vector* $\boldsymbol{b}$*, an MBR combination* p *is* dominated *by an object combination* s *if its upper bound* $p^\top$ *is dominated by* s*, namely,* $d_{p^\top}^k < d_s^k$ $(k \in 1, 2, \cdots, m)$ *and* $d_{p^\top}^j \leq d_s^j$ $(j \in 1, 2, \cdots, m$ and $j \neq k)$*.* □

Theorem 2. *An MBR combination* p *cannot be expanded to optimal object combinations if it is dominated by a combination* s*.*

Proof 2. *An MBR combination* p *with an upper bound* $p^\top$ *can be expanded to an object combination* p' *which have upper bounds within* $p^\top$*, namely,* $p'^i \leq p^{i\top}$ $(i \in 1, 2, \cdots, m)$*. If* p *is dominated by an object combination* s*,* p' *is also dominated by* s *because* $d_{p'}^k < d_s^k$ $(k \in 1, 2, \cdots, m)$ *and* $d_{p'}^j \leq d_s^j$ $(j \in 1, 2, \cdots, m$ and $j \neq k)$*.* ■

Example 5. *Let us consider the upper bound reduction process in Fig. 4 (a). The upper bounds of* $\{abb\}$*,* $\{aab\}$*,* $\{aaa\}$ *and* $\{bbb\}$ *are shown the figure. At the leaf level, we have obtained object combinations (i.e.,* $\{BAC\}, \ldots, \{FCE\}$*) by expanding the MBR combination* $\{abb\}$*. Their attributes are shown in the figure. Considering the Theorem 2, the MBR combination* $\{bbb\}$ *is non-promising to generate optimal combinations because its upper bound* $(12, 15)$ *is dominated by* $\{DCE\} = (12, 16)$ *already found.* ■

We use a min-heap to organize the MBR combinations which are waiting to be expanded like the well-known BBS algorithm in [8]. Each time we pop and expand the top one and then push its expansions into the min-heap. The top one should have a minimum *Manhattan distance* to an objective vector $\boldsymbol{b}$.

Definition 6 (Manhattan Distance of An MBR Combination). *Given an MBR combination* p *with lower bounds* $p^{i\perp} \leq b^i$ $(i \in 1, 2, \cdots, m)$*, the* Manhattan distance *of the combination* p *is* $md(p) = \Sigma_{j=1}^{r} d^{j\top}$ *where* $d^{j\top} = b^j - p^{j\top}$ $(p^{j\top} \leq b^j$ *and* $r \leq m)$*.* □

For example, an MBR combination $\{bbb\}$ has a lower bound $(6, 9)$ and an upper bound $(12, 15)$. Its lower bound and upper bound are within the objective vector $\boldsymbol{b} = (13, 16)$. Its Manhattan distance is $md(\{bbb\}) = (13 - 12) + (16 - 15) = 2$. Notice that MBR combinations like $\{aab\}$ are special. The MBR combination $\{aab\}$ has a lower bound $(12, 15)$ within $(13, 16)$ but an upper bound $(16, 23)$ beyond of $(13, 16)$. We set its Manhattan distance to $md(\{aab\}) = 0$.

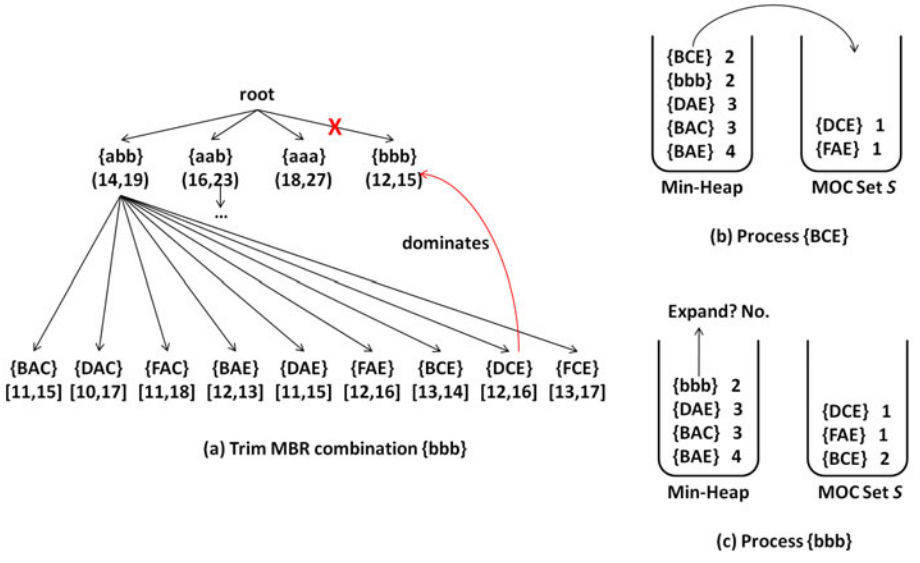


Fig. 4. Upper Bound Reduction

Theorem 3. *An object combination s, which is extended from an MBR combination p, cannot dominate another MBR combination p', if p' has a smaller or equal Manhattan distance with p, namely, $md(p') \leq md(p)$.*

Proof 3. *The object combination s has a Manhattan distance $md(s) = \Sigma_{i=1}^{m} d_s^i$ where d_s^i is its distance to $\boldsymbol{b}$ at the i^{th} attribute. The MBR combination p, where s comes from, has a Manhattan distance $md(p) \leq md(s)$. Assume that the object combination s can dominate another MBR combination p', say, $d_s^k < d_{p'\top}^k$ $(k \in 1, 2, \cdots, m)$ and $d_s^j \leq d_{p'\top}^j$ $(j \in 1, 2, \cdots, m$ and $j \neq k)$. Then p' has a Manhattan distance $md(p') > md(p)$ because $md(p') > md(s)$ and $md(p) \leq md(s)$. It contradicts with the condition $md(p') \leq md(p)$ in Theorem 3.* ■

According to Theorem 3, we maintain a min-heap with respect to Manhattan distances of combinations. The top one has a minimum Manhattan distance. Other combinations in the heap cannot generate a combination which will dominate the top one. Each time we pop and expend the top one to new combinations. If a new combination is an object combination, we update the current optimal combination set S. If a new combination is still an MBR combination, we decide whether it is dominated by current optimal combinations in S. We only push the non-dominated ones into the min-heap and throw away the dominated ones. After rebuilding the min-heap, we pop a new top one and expand it by repeating the process stated above until the min-heap is empty. Notice that if a top one is dominated by any current optimal combination in S, we throw it away directly.

Example 6. *Fig. 4 (b) shows the scene when an combination $\{BCE\}$ is on the top of the min-heap. The number shown behind of each combination is its*

Manhattan distance. The combination $\{BCE\}$ *is an object combination and we compare it with two optimal combinations* $\{DCE\}$ *and* $\{FAE\}$ *which have been found already. It cannot be dominated neither by* $\{DCE\}$ *nor* $\{FAE\}$*. We pop out combination* $\{BCE\}$ *and inserted it into the MOC set* S*.*

After popping out combination $\{BCE\}$*, an MBR combination* $\{bbb\}$ *is on the top of the min-heap as Fig. 4 (c) shows. We pop out* $\{bbb\}$ *but do not expand* $\{bbb\}$ *because it is dominated by* $\{DCE\} \in S$*.* ■

Algorithm 2 shows an MOC query using the lower bound reduction as well as the upper bound reduction. Algorithm 2 is similar to Algorithm 1 except for several differences annotated by comments. In the beginning, we decide whether a combination p is dominated by combinations in S using a function is_domed(p, S) (line 2). If it cannot be dominated, the process continues. We update S if p is an object combination (line 4 to 5). We expand p if it is an MBR combination (line 7 to line 38). If a complete instantiated combination p' cannot be dominated by combinations in S, we calculate its Manhattan distance $md(p')$ using a function calculate_md(p′) and push it into the min-heap Q (line 32 to 36). Note that the min-heap rebuild itself after push or pop operations. When the min-heap Q is not empty, we pop and use a new top one to execute the function MOC_query().

4 Experiments

We implemented Algorithm 2 in GNU C++ and conducted experiments on an Intel Core2 Duo 2.40 GHz PC (2.0 GB RAM) with a Fedora 12 Linux 2.6.32. The algorithm was implemented based on the R-tree provided by a spatial index library SaIL ([9,10]). The R-tree has a block size 512 bytes and a fill factor 70%.

We evaluated performances of Algorithm 2 with four experimental sets. The first set evaluated the algorithm with respect to different data distributions, say, independent distribution, correlated distribution, and anti-correlated distribution. The second set evaluated the algorithm with different sizes of data sets. The third set evaluated the algorithm with respect to different m's where m is the number of attributes. The fourth set evaluated the algorithm with respect to different cardinalities h's where h is the number of objects in a combination. We will show the experimental results of the three sets in Section 4.1, Section 4.2, Section 4.3, and Section 4.4 respectively.

4.1 Performances on Different Data Distributions

When we evaluate algorithm performances with different data distributions, we use five synthetic data sets $D_{-0.6}$, $D_{-0.4}$, D_0, $D_{0.4}$ and $D_{0.6}$ with different correlation coefficients -0.6, -0.4, 0.0, 0.4 and 0.6. We generated these data sets using the method in [2]. Objects in data sets $D_{0.4}$ and $D_{0.6}$ follow the correlated distribution while object in data sets $D_{-0.4}$ and $D_{-0.6}$ follow the anti-correlated distribution. Objects in the data set D_0 follows the uniform distribution. Each data set has 10K objects with two attributes ranging from 0 to 10000. We randomly select 50 different objective vectors ranging in $[1000, 9000] \times [1000, 9000]$

Algorithm 2. MOC Query Using Lower Bound Reduction and Upper Bound Reduction

```
1: procedure MOC_query(p, b, h, S, Q)                    {Q is the min-heap}
2: if is_domed(p, S) then                                {p is dominated by combinations in S.}
3:     return;
4: if is_leaf_combination(p) then                        {p is an object combination.}
5:     update_optimal_set(p, S);
6: else                                                  {Expand an MBR combination p.}
7:     begin
8:     p' := v1 v2 ··· vh;
9:     T := b;
10:    for i := 1 to h do
11:        d1i := get_children(ei);
12:    l := 1;
13:    while true do
14:        begin
15:        if dll ≠ ∅ then
16:            vl := get_MBR(dll);
17:        else
18:            if l = 1 then
19:                break;
20:            else
21:                begin
22:                l := l − 1;
23:                continue;
24:                end
25:        if l < h then
26:            begin
27:            T := T − vl⊥;
28:            forward_check(T, l, i);
29:            l := l + 1;
30:            end
31:        else
32:            if ¬is_domed(p', S) then
33:                begin
34:                calculate_md(p');                     {Calculate Manhattan distance of p'.}
35:                push(Q, p');                          {Push the new combination p' into Q.}
36:                end
37:        end
38:    end
39: if Q ≠ ∅ then
40:    begin
41:    p = pop(Q);                                       {Pop the top combination.}
42:    MOC(p, b, h, S, Q);                               {Use p to do a new MOC query.}
43:    end
```

to evaluate the algorithm. After executing queries to find out 3-MOCs on these five data sets, we summarized average results of the random 50 different queries as OC which is the number of optimal combinations; CMC which is the number

of checked MBR combinations; CAD which is the number of candidate object combinations for optimal ones; CPU which is the cost of running time with one second as a unit.

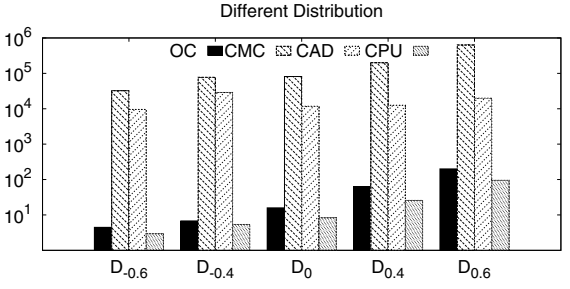


Fig. 5. Performances on Different Data Distribution

Fig. 5 shows the experimental results of data sets with different object distributions. Note that OC, CMC, CAD and CPU are all in their log scales. The correlated data sets (i.e. $D_{0.6}$ and $D_{0.4}$) have more OCs than the anti-correlated data sets (i.e. $D_{-0.6}$ and $D_{-0.4}$). The uniform distribution data set (i.e. D_0) has a middle size OCs. The number of candidate object combinations CAN is not influenced by the distributions of data sets. The CPU cost depends on how many MBR combinations (CMC) we have checked during the MOC queries. We have to check more CMCs for the correlated data sets while check fewer CMCs for the anti-correlated data sets.

4.2 Performances on Different Data Sizes

When we evaluate algorithm performances with different data sizes, we use five synthetic data sets D_{1K}, D_{2K}, D_{5K}, D_{10K} and D_{15K} containing 1K objects, 2K objects, 5K objects, 10K objects and 15K objects respectively. Objects in each data set have two attributes and follow a uniform distribution. We also randomly select 50 different objective vectors to evaluate the algorithm. After executing queries to find out 3-MOCs on these five different data sets, we summarized average results of the random 50 different queries as OC, CMC, CAD and CPU in Fig. 6. Note that OC, CMC, CAD and CPU are all in their log scales.

The data set (e.g. D_{15K}) containing more objects has more OCs than the data set (e.g. D_{1K}) containing fewer objects. The number of candidates (CAN) of optimal combinations is not influenced by the sizes of data sets. We have to check more MBR combinations (CMC) for a large data set (e.g. D_{15K}) than for a small data set (e.g. D_{1K}). The CPU cost depends on the CMC and it increases with the growth of the data set size.

4.3 Performances on Different Numbers of Attributes

When we evaluate algorithm performances with different attribute number m, we use three data sets D_2, D_3 and D_4 where $m = 2$, $m = 3$ and $m = 4$

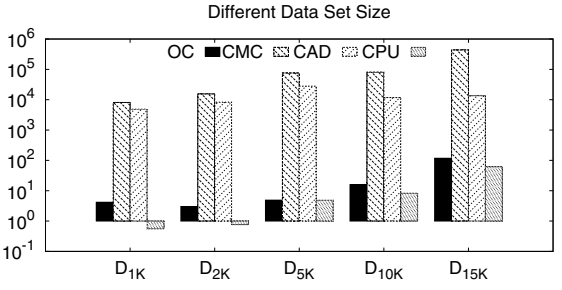


Fig. 6. Performances on Different Data Sizes

respectively. The objects in the three data sets follow uniform distributions. Each data set contains 100 objects with attribute values ranging from 0 to 1000. We use 15 objective vectors $\boldsymbol{b_i}$ $(i \in 1, 2, \cdots, 15)$ where $b_i^1 = b_i^2 = \cdots = b_i^m = 400 + 200 \times i$ $(m = 2, 3, 4)$. Given the objective vector $\boldsymbol{b_i}$, we execute MOC queries on D_2, D_3 and D_4 in order to find out optimal combinations consisting of 3 objects.

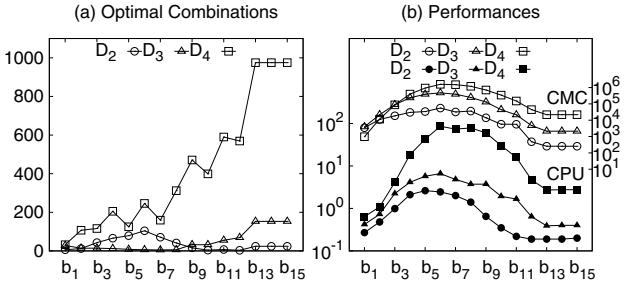


Fig. 7. Performances on Data Sets D_2, D_3 and D_4

Fig. 7 (a) shows the number of optimal combinations on data sets D_2, D_3 and D_4. The vertical axis represents the number and the horizontal axis represents objective vectors $\boldsymbol{b_1}$ to $\boldsymbol{b_{15}}$. The data set with a larger m (e.g. D_4) has more optimal combinations than the data set with a smaller m (e.g. D_2) because it is difficult for one combination dominates another combination if there are more attributes to compare.

The Fig. 7 (b) shows the algorithm performances on data sets D_2, D_3 and D_4. The left vertical axis represents CPU cost with a second unit in a log scale while the right vertical axis represents the number of CMCs also in a log scale. The CPU cost depends on the number of CMCs. The data set with a larger m (e.g. D_4) checks more MBR combinations than the data set with a smaller m (e.g. D_2) because the R-tree has more MBRs in a high-dimensional space.

4.4 Performances on Different Cardinality

When we evaluate algorithm performances on different cardinalities of a combination, say, different h's, we use the uniform distribution data set D_{1K}. Given the objective vector $\boldsymbol{b} = (500, 500)$, we execute MOC queries to find out optimal combinations with cardinalities $h = 1, 2, \cdots, 9$.

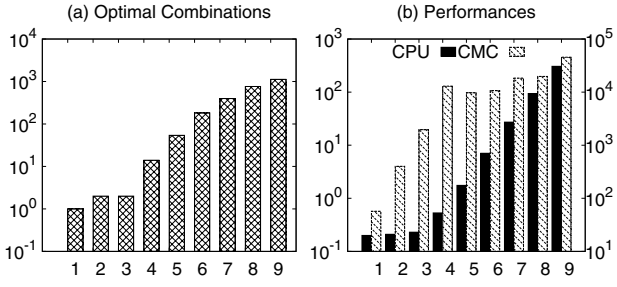


Fig. 8. Performances on Data Set D_{1k} with $h = 1, 2, \cdots, 9$

Fig. 8 (a) shows the number of optimal combinations with different h's. The horizontal axis represents the h from 1 to 9 and the vertical axis represents the number in a log scale. The number increases while h increases because a same object set can generate more object combinations with a larger cardinality (e.g. $h = 9$).

Fig. 8 (b) shows the algorithm performances with different h's. The left vertical axis represents the CPU cost while the right vertical axis represents the number of CMCs. The CPU cost depends on the number of CMCs as well as the number of candidates. The number of CMC grows with h because a same R-tree can generate more MBR combinations which have a larger cardinality (e.g. $h = 9$). At the leaf level of the R-tree, we decide whether a popped candidate object combination is an optimal one. It takes much more time to do dominance tests for a larger number of candidates due to a larger cardinality h.

5 Conclusions

In this paper, we propose a new multi-objective optimization problem called MOC problem. The MOC problem is to find out optimal combinations consisting of h objects with respect to a given objective vector $\boldsymbol{b}$. The optimal combinations cannot be dominated by any possible combinations. We organize objects using the R-tree index and do MOC queries efficiently with two reduction methods, say, the lower bound reduction and the upper bound reduction. We evaluated the proposed MOC query algorithm on different data sets with different objective vectors and parameter settings.

Acknowledgments. This research was partly supported by Grant-in-Aids for Scientific Research (#22300034) from JSPS and the Funding Program for World-Leading Innovative R&D on Science and Technology (First Program).

References

1. Deb, K.: Multi-objective optimization using evolutionary algorithms, pp. 13–46. John Wiley and Sons, Chichester (2001)
2. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: ICDE, pp. 421–430 (2001)
3. Su, I.-F., Chung, Y.-C., Lee, C.: Top-k combinatorial skyline queries. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5982, pp. 79–93. Springer, Heidelberg (2010)
4. Roy, S.B., Yahia, S.A., Chawla, A., Das, G., Yu, C.: Constructing and Exploring Composite Items. In: SIGMOD, pp. 843–854 (2010)
5. Papadias, D., Mamoulis, N., Delis, V.: Algorithms for Querying by Spatial Structure. In: VLDB, pp. 546–557 (1998)
6. Bertsimas, D., Tsitsiklis, J.N.: Introduction to Linear Optimization, pp. 451–531 (1997)
7. Papadias, D., Mamoulis, N., Delis, V.: Algorithms for Querying by Spatial Structure. In: VLDB, pp. 546–557 (1998)
8. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1), 41–82 (2005)
9. Hadjieleftheriou, M., Hoel, E., Tsotras, V.J.: SaIL: A Spatial Index Library for Efficient Application Integration. Geoinformatica 9(4), 367–389 (2005)
10. Hadjieleftheriou, M.: Spatial Index Library (SaIL), http://www2.research.att.com/~marioh/spatialindex/
11. Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching. In: SIGMOD, pp. 47–57 (1984)
12. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: ICDE, pp. 717–719 (2003)
13. Ryan, P.G., Shipley, R., Gryz, J.: Maximal Vector Computation in Large Data Sets. In: VLDB, pp. 229–240 (2005)
14. Wikipedia, "Knapsack Problem", http://en.wikipedia.org/wiki/Knapsack_problem

NEFOS: Rapid Cache-Aware Range Query Processing with Probabilistic Guarantees

Spyros Sioutas[1], Kostas Tsichlas[2], Ioannis Karydis[1], Yannis Manolopoulos[2], and Yannis Theodoridis[3]

[1] Department of Informatics, Ionian University, Greece
(sioutas,karydis)@ionio.gr
[2] Department of Informatics, Aristotle University of Thessaloniki, Greece
(tsichlas,manolopo)@csd.auth.gr
[3] Department of Informatics, University of Piraeus, Greece
ytheod@unipi.gr

Abstract. We present NEFOS (NEsted FOrest of balanced treeS), a new cache-aware indexing scheme that supports insertions and deletions in $O(1)$ worst-case block transfers for rebalancing operations (given and update position) and searching in $O(\log_B \log n)$ expected block transfers, (B= disk block size and n= number of stored elements). The expected search bound holds with high probability for any (unknown) *realistic* input distribution. Our expected search bound constitutes an improvement over the $O(\log_B \log n)$ expected bound for search achieved by the ISB-tree (Interpolation Search B-tree), since the latter holds with high probability for the class of *smooth* only input distributions. We define any unknown distribution as realistic if the smoothness doesn't appear in the whole data set, still it may appear locally in small spatial neighborhoods. This holds for a variety of real-life non-smooth distributions like skew, zipfian, powlaw, beta e.t.c.. The latter is also verified by an accompanying experimental study. Moreover, NEFOS is a B-parametrized concrete structure, which works for both I/O and RAM model, without any kind of transformation or adaptation. Also, it is the first time an expected sub-logarithmic bound for search operation was achieved for a broad family of non-smooth input distributions.

Keywords: Data Structures, Data Management Algorithms.

1 Introduction

More than three decades after its invention, B-tree [5] and its variants remain the ubiquitous external memory data structure for indexing and organizing large data sets with numerous applications, especially in database systems. Its popularity is mainly due to the $O(\log_B n)$ worst-case complexity (block transfers) for search and update operations. The most heavily used application is the efficient answering of one-dimensional range search queries using $O(\log_B n + r)$ block transfers, where $R = rB$ is the number of elements reported, B is the block-size and n the number of element. In this paper, we consider one of the most known

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 62–77, 2011.

and widely used such models, namely the two-level memory hierarchy model introduced in [2,25]. In this model, the memory hierarchy consists of an internal (main) memory and an arbitrarily large external memory (disk) partitioned into blocks of size B. The data from the external to the main memory and vice versa are transferred in blocks (one block at a time).

A large number of variants of the B-tree have been proposed since its appearance in order to improve its performance in practice for various applications — see the excellent survey by Vitter [24] for an extended accounting of these and other variants and their applications — to make it parallel for use in multi-disk environments [21], to tune it for concurrency and recovery purposes [14,22], to extend it to cover other than the original field [9], etc. Regarding the update operation, it should be noted that an update operation consists of three consecutive phases: a search phase (to locate the place of the update), an element-updating phase (to insert the new element, or delete the located element), and a rebalancing phase (to restore the B-tree structure). Excluding the first phase (search operation), the dominating phase of an update operation is the rebalancing one, since the element-updating phase takes typically $O(1)$ block transfers (and/or time). In the case of B-tree and its variants, the rebalancing phase requires $\Theta(\log_B n)$ block transfers in the worst-case. This implies that the update operation takes $\Theta(\log_B n)$ block transfers, even in the case where the update position (block within which the update will take place) is given.

ISB-tree (Interpolation Search B-tree) presented in [12], supports search operations in $O(\log_B \log n)$ expected block transfers *with high probability* (w.h.p.) for a large class of input distributions (including both uniform and non-uniform classes) described below, and update operations in $O(1)$ block transfers, provided that the update position is given. The search bound implies that a one-dimensional range search query can be supported in $O(\log_B \log n + r)$ expected block transfers with high probability. The worst-case block transfers for the search operation are $O(\log_B n)$.

The expected search bound was achieved by considering a rather general scenario of μ-*random* insertions and *random* deletions, where μ is a so-called *smooth* probability density [3,19]. An insertion is μ-random if the key to be inserted is drawn randomly with density function μ; a deletion is random if every key present in the data structure is equally likely to be deleted [13]. Informally, a distribution defined over an interval I is *smooth* if the probability density over any subinterval of I does not exceed a specific bound, however small this subinterval is (i.e., the distribution does not contain sharp peaks). Smooth distributions are a superset of uniform, bounded, and several non-uniform distributions (e.g., the class of regular distributions introduced by Willard [26]).

In this paper, we present NEFOS (NEsted FOrest of balanced treeS), a new cache-aware indexing scheme that supports insertions and deletions in $O(1)$ worst-case block transfers for rebalancing operations (provided that the update position is given) and searching in $O(\log_B \log n)$ expected block transfers, where B represents the disk block size and n denotes the number of stored elements. The expected search bound holds with high probability for any (unknown)

realistic input distribution. Our expected search bound constitutes an improvement over the $O(\log_B \log n)$ expected bound for search achieved by the ISB-tree (Interpolation Search B-tree), since the latter holds with high probability for the class of smooth only input distributions. Note here that *realistic* distributions are a superset of smooth distributions. Generally speaking, in $(f1, f2)$-smooth distributions, $f1$ measures how fine is the partitioning of an arbitrary subinterval as well as $f2$ measures the sparseness of this subinterval. In this context, any probability distribution is $(f1, \Theta(n))$-smooth. For smooth class of densities, $f2 = \Theta(n^{\delta})$, where $0 < \delta < 1$. We define any unknown distribution as realistic if there is at least one subinterval of $\Theta(n)$ sparseness and there are at least $\Theta(n^{\delta})$ consecutive subintervals of $\Theta(n^{1-\delta})$ sparseness, where $0 < \delta < 1$. This holds for a variety of real-life non-smooth distributions like skew, zipfian, powlaw, beta e.t.c., where the *smoothness* property appears locally in small spatial neighborhoods. The latter is also verified by an accompanying experimental study. Moreover, NEFOS is a B-parametrized concrete structure, which works for both I/O (arbitrary B) and RAM (B=2) model, without any transformation or adaptation. Also, it is the first time an expected sub-logarithmic bound for search operation was achieved for a broad family of non-smooth input distributions.

External data structures related to our approach are those based on hashing [18,24]. The main representatives of external memory hashing methods include: extendible hashing [8], linear hashing [16], and external perfect hashing [10]. These hashing schemes and their variants need $O(1)$ expected block transfers for answering search queries, but they share various disadvantages when compared to our structure: (i) they do not support range queries; (ii) their expected case analysis usually assumes *uniform* input distributions (or input distributions that produce uniform hash key values); and (iii) they exhibit poor worst case performance.

The remainder of the paper is organized as follows. In Section 2, we discuss preliminary notions and results that are used throughout the paper, define formally the class of smooth probability distributions as well as discuss the *ISB-Tree*. The main result of this paper, the *NEFOS-tree*, with the complexity analysis of its operations is discussed in Section 3. Section 4 provides an experimental evaluation with synthetic and real data of our theoretical findings. We conclude in Section 5.

2 Preliminaries

This Section briefly describes B-trees, input distributions in the context of internal memory data structures as well as the static interpolation search tree.

The B-tree. The *B-tree* is a $\Theta(B)$-ary tree (with the root possibly having smaller degree) built on top of $\Theta(n/B)$ leaves. The degree of internal nodes, as well as the number of elements in a leaf, is typically kept in the range $[B/2, B]$ such that a node or leaf can be stored in one disk block. All leaves are on the same level and the tree has height $O(\log_B n)$. This guarantees that a search operation can be accomplished within $O(\log_B n)$ block transfers. An insertion is performed

in $O(\log_B n)$ block transfers by first searching down the tree for the relevant leaf l. The insertion there may cause a split and the latter may propagate up the tree. Similarly, a deletion can be performed in $O(\log_B n)$ block transfers by first searching down the tree for the relevant leaf l and then removing the deleted element. The deletion there may cause a fusion and the latter may propagate up the tree.

The Lazy B-tree. The Lazy B-tree of [12] is a simple but non-trivial externalization of the techniques introduced in [20]. The first level consists of an ordinary B-tree, whereas the second one consists of buckets of size $O(\log^2 n)$, where n is approximately equal to the number of elements stored in the access method. The following theorem provides the complexities of the Lazy B-tree:

Theorem 1. The Lazy B-Tree supports the search operation in $O(\log_B n)$ worst-case block transfers and update operations in $O(1)$ worst-case block transfers, provided that the update position is given.

Proof. see [12].

The ISB-tree. The ISB-tree is a two-level data structure. The upper level is a non - straightforward externalization of the Static Interpolation Search Tree (SIST) presented in [11]. In the definition of the (f_1, f_2)-smooth densities [26,19], intuitively, function f_1 partitions an arbitrary subinterval $[c_1, c_3] \subseteq [a, b]$ into f_1 equal parts, each of length $\frac{c_3 - c_1}{f_1} = O(\frac{1}{f_1})$; that is, f_1 measures how fine is the partitioning of an arbitrary subinterval. Function f_2 guarantees that no part, of the f_1 possible, gets more probability mass than $\frac{\beta \cdot f_2}{n}$; that is, f_2 measures the sparseness of any subinterval $[c_2 - \frac{c_3 - c_1}{f_1}, c_2] \subseteq [c_1, c_3]$. The class of (f_1, f_2)-smooth distributions (for appropriate choices of f_1 and f_2) is a superset of both regular and uniform classes of distributions, as well as of several non-uniform classes [3,11]. Actually, *any* probability distribution is $(f_1, \Theta(n))$-smooth, for a suitable choice of β. The following theorem presented in [12] follows and holds for the very broad class of $(n/(\log\log n)^{1+\epsilon}, n^{1-\delta})$-smooth densities, where $\delta = 1 - \frac{1}{B}$ and includes the uniform, regular, bounded as well as several non-uniform distributions.

Theorem 2. Suppose that the upper level of the ISB-tree is an external static interpolation search tree with parameters $R(s_0) = s_0^{\delta}$, $I(s_0) = s_0/(\log\log s_0)^{1+\epsilon}$, where $\epsilon > 0$, $\delta = 1 - \frac{1}{B}$, $s_0 = n_0$, n_0 is the number of elements in the latest reconstruction, and that the lower level is implemented as a forest of Lazy B-trees. Then, the ISB-tree supports search operations in $O(\log_B \log n)$ expected block transfers with high probability, where n denotes the current number of elements, and update operations in $O(1)$ worst-case block transfers, if the update position is given. The worst-case update bound is $O(\log_B n)$ block transfers, and the structure occupies $O(n/B)$ blocks.

Proof. see [12].

3 NEFOS

In the following we present the building phase of NEFOS as well as the complexity analysis of its basic operations.

3.1 Building NEFOS

Let $\mu(\cdot)$ random be the sequence of inserted keys and random be the sequence of deleted keys. Let n be the total number of w-bit keys, which are organized in block fashion. Let $n_1 = O(n/B)$ the number of these blocks. Let also $n_2 = O(n_1/log_B log_B n)$ super-blocks each of which contains $O(log_B log_B n)$ blocks. Let also $\mu_1(\cdot)$ random be the sequence of keys of super-block representatives. According to combinatorial game of bins and balls presented in [11], we store these keys in $N = n_2/lnn_2$ buckets, each of which contains $O(lnn_2)$ keys.

Lemma 1. If the sequence of inserted keys remains $\mu(\cdot)$ random then the sequence of super-block representative keys remains $\mu_1(\cdot)$ random.

Proof. See [11].

Lemma 2. Given a $\mu_1(\cdot)$ random sequence of inserted super-block representative keys and a random sequence of deleted super-block representative keys, the load of each bucket never becomes zero and never exceeds $\Theta(\text{polylog } N)$ keys in expected w.h.p. case.

Proof. See [11].
In $(f1, f2)$-smooth distributions, $f1$ measures how fine is the partitioning of an arbitrary subinterval and $f2$ measures the sparseness of this subinterval. In this context, any probability distribution is $(f1, \Theta(n))$-smooth. In any realistic distribution there are sparse subintervals of $\Theta(n)$ sparseness and dense subintervals of $\Theta(n^{1-\delta})$ sparseness, where $0 < \delta < 1$. For example in Figure 1, we depict with red color a sparse subinterval and with blue color a number of consecutive dense subintervals.

In the same context and according to Lemmas 1 and 2 we construct sparse and dense buckets of polylogaritmic load. For example see the red and blue buckets of Figure 2.

Let $bucket_{I_{max}}$ the bucket with the maximum range of keys (I_{max}) and $bucket_{I_{min}}$ the bucket with the minimum range of keys (I_{min}). For any realistic distributions $I_{max} = \Theta(n)$ and $I_{min} = \Theta(n^{1-\delta})$, where $0 < \delta < 1$.

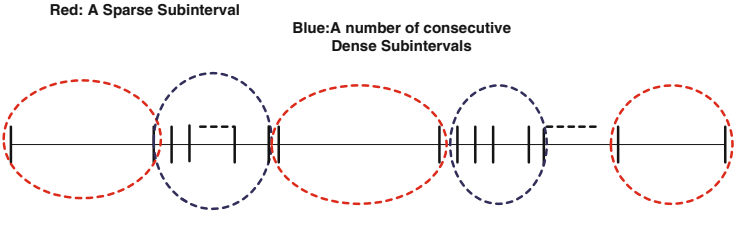


Fig. 1. Sparse and dense subintervals of any arbitrary distribution

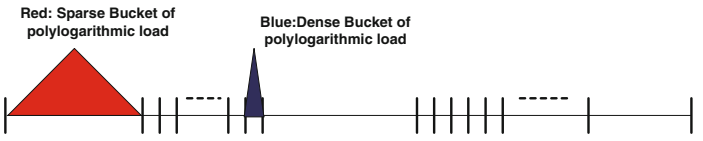


Fig. 2. Red and Blue Buckets

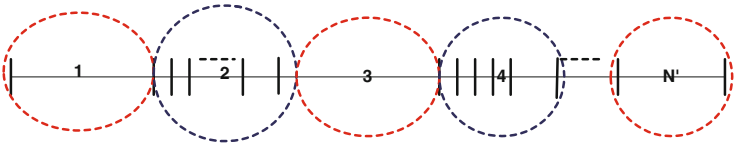


Fig. 3. Red and Blue Labeled Cluster_Nodes. Blue Cluster_node contains at least one dense subinterval.

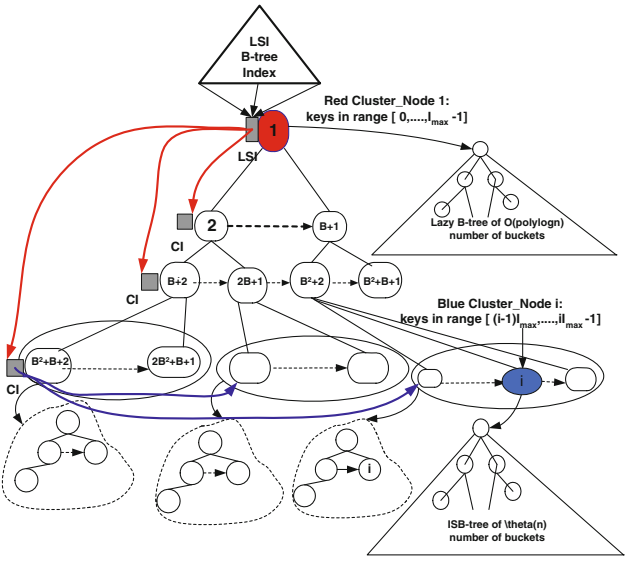


Fig. 4. NEsted FOrest of load-balancing treeS in I/O model

Now, we build labeled cluster_nodes. Each cluster_node with label i' (where $1 \leq i' \leq N'$) stores ordered buckets with keys belonging in the range $[(i'-1)I_{max}, \ldots, i'I_{max} - 1]$, where N' is the number of cluster_nodes (see Figure 3).

NEFOS stores cluster_nodes only, each of which is structured either as a Lazy B-tree if its color is red or as an ISB-tree if its color is blue. In particular, NEFOS is built by grouping cluster_nodes having the same ancestor and organizing them in a tree structure recursively. The innermost level of nesting (recursion) will be characterized by having a tree in which no more than B cluster_nodes share the same direct ancestor, where B is the disk block. Thus, multiple independent trees are imposed on the collection of nodes (see Figure 4).

The degree of the nodes at level $i > 0$ is $d(i) = t(i)$, where $t(i)$ indicates the number of nodes at level i. It is defined that $d(0)$=B and $t(0)$=1. It is apparent that $t(i) = t(i-1)d(i-1)$, and, thus, by putting together the various components, we can solve the recurrence and obtain $d(i) = t(i) = B^{2^{i-1}}$ for $i \geq 1$. We stope at the level where each collection contains $O(N'^{1/B})$ cluster_nodes. For example in figure 4, the root is located at level 0, thus the funout of root is exactly B. In particular the B cluster_nodes located at level 1 have labels 2, 3, $\ldots B+1$ respectively. At level 1, the B labeled nodes $B+2,\ldots, 2B+1$ rooted at labeled node 2, the B labeled nodes $2B+2,\ldots, 3B+1$ rooted at labeled node 3, e.t.c. and the B labeled nodes $B^2+2,\ldots, B^2+B+1$ rooted at labeled node $B+1$. At level 2, the funout is B^2, so the B^2 labeled nodes $B^2+B+2,\ldots, 2B^2+B+1$ rooted at labeled node $B+2$ and so on.

We also equip the root cluster_node with a table named *Left Spine Index* (LSI), which stores pointers to the cluster_nodes of the left-most spine. We organize LSI table as a B-tree. For example in figure 4, see the red pointers beginning from cluster_node 1 towards cluster_nodes with labels 2, $B+2$ and B^2+B+2 respectively.

Furthermore, each cluster_node of the left-most spine is equipped with a table named *Collection Index* (CI), which stores pointers to the collections of cluster_nodes presented at the same level. Cluster_nodes having the same father belong to the same collection. We also organize CI tables in block fashion. For example in figure 4, see the blue pointers beginning from the left-most collection towards the other collections located at the same level.

Finally, each cluster_node is organized in a Lazy B-tree manner and each collection is organized in a NEFOS manner at the next level of nesting (see the dash lines in figure 4).

Remark 1. If we parametrize B and choose such a small value (f.e. B=2) so as the whole structure can fit in main memory as well as replace each lazy B-tree and the B-tree of LSI structure with q*-heap machinery [27], then NEFOS becomes a data structure in RAM model with the same expected complexities w.h.p. for all operations, without any kind of transformation or adaptation.

3.2 Complexity Analysis

We will focus first on space and then on time complexity of NEFOS' basic operations.

Space Complexity Analysis. The double exponentially increasing fanout guarantees the following lemma:

Lemma 3. The height (or the number of levels) of NEFOS is $O(\log\log_B n)$ in the worst case.

Proof. It is obvious if we solve for i the equation $B^{2^{i-1}} = O(N'^{1/B})$.

Now, since the innermost level of nesting (recursion) is characterized by having a tree in which no more than B cluster_nodes share the same direct ancestor, the lemma 4 follows:

Lemma 4. The maximum number of possible nestings in NEFOS structure is $O(\log_B \log_B n)$ in the worst case.

Proof. It is obvious that in NEFOS structure of j^{th} nested level, the last collections contain $O(N'^{1/B^j})$ cluster_nodes. Since the innermost level of nesting (recursion) is characterized by having a tree in which no more than B cluster_nodes share the same direct ancestor, it holds that $O(N'^{1/B^j}) = B$, meaning that $j = O(\log_B \log_B N')$ or $j = O(\log_B \log_B n)$.

Finally, the sizes of CI and LSI tables are described by the following lemma:

Lemma 5. The maximum size of the CI and LSI tables is $O(\frac{n^{1/B}}{B})$ and $O(\frac{\log \log_B n}{B})$ in worst-case respectively.

Proof. Since the maximum number $(O(n^{1/B}))$ of cluster_nodes appears at last level of the basic (non-nested) NEFOS structure, the length of CI is $O(n^{1/B})$. Since, CI has been organized in a block fashion,the $O(\frac{n^{1/B}}{B})$ space complexity follows. The length of LSI table depends on the height of NEFOS. Thus according to lemma3, this length becomes $O(\log \log_B n)$. Since, LSI has been organized in a block fashion (according to B-tree),the $O(\frac{\log \log_B n}{B})$ space complexity follows.

Each cluster_node appears in $O(\log_B \log_B n)$ nesting levels in the worst case. As a result each bucket appears in $O(\log_B \log_B n)$ nesting levels in the worst case and as a result each super-block appears in $O(\log_B \log_B n)$ nesting levels in the worst case. Since, each super-block contains $O(log_B log_B n)$ blocks, we have $O(n/B)$ blocks in total and the theorem foloows:

Theorem 3. The whole space of NEFOS remains linear.

Query Processing, Data Insertion, Data Deletion. Assume we are located at root cluster_node and seek a key k. First, we find the range where k belongs in. Let say $k \in [(j-1)\ I_{max}, jI_{max} - 1]$. The latter means that we have to search for cluster_node j. The first step of our algorithm is to find the level where the desired cluster_node j is located. For this purpose, we organize the cluster_node labels pointed by the LSI table in a B-tree manner (see Figure 4). Since, according to lemma 5, the maximum size of the LSI table is $O(\frac{\log \log_B n}{B})$ in worst-case, the theorem 4 follows.

Theorem 4. The level where the desired cluster_node j is located can be found out in $O(\log_B(\frac{\log \log_B n}{B}))$ I/Os.

Let say that cluster_node j is located at the i-th level. We follow the i-th pointer of the LSI table located at root cluster_node so as to reach the leftmost cluster_node x of level i. Then, we compute the collection in which the cluster_node j belongs. Since the number of collections at level i equals the number of cluster_nodes located at level $(i-1)$, we divide the distance between j and x by the factor $t(i-1)$. Let m (in particular $m = \left\lceil \frac{j-x+1}{t(i-1)} \right\rceil$) be the result of this division. The latter means that we additionally need $O(1)$ I/Os to follow the $(m+1)$-th pointer of the CI table so as to reach the desired collection. Since the

collection indicated by the CI[m+1] pointer is organized in the same way at the next nesting level, we continue this process recursively.

Generally speaking, we need $O(\log_B(\frac{\log\log_B n}{B})) + O(1)$ I/Os for locating the desired collection and we have to continue this process recursively for all nesting levels. Since the maximum number of nesting levels is $O(\log_B \log_B n)$ in the worst case (according to lemma 4), the whole searching process requires $T_1(n)$ I/Os to locate the target cluster_node, where:

$$T_1(n) = \sum_{i=1}^{\log_B \log_B n} \log_B(\frac{\log\log_B n^{\frac{1}{B^{i-1}}}}{B}) \quad (1)$$

from which we get:

$$T_1(n) < O(\log_B \log n)$$

Then, we have to locate the target bucket by searching the respective Lazy B-tree or ISB-tree, requiring $T_2(n)$ I/Os.

If the located cluster_node is red, then its load is polylogarithmic and the lazy B-tree index is sufficient to give us the desired complexity. If it's blue then its load may be $\Theta(n)$, and the question is how we can compute the new maximum range of keys there. Let say it $I_{max(1)}$ (see the Figure 5).

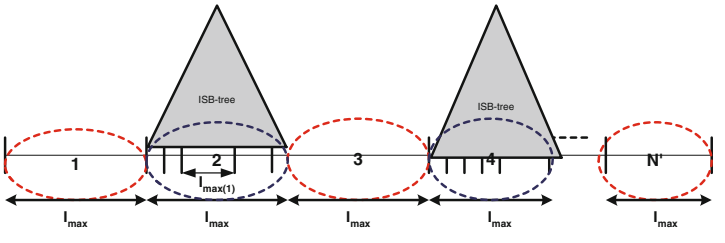


Fig. 5. Blue cluster_nodes are structured in NEFOS manner with new range Imax(1)

Since each blue cluster_node contains at least one dense subinterval of $\Theta(n^{1-\delta})$ sparseness, where $0 < \delta < 1$, for the new range $I_{max(1)}$ the following holds:

$I_{max(1)} = I_{max} - f(n) \cdot \Theta(n^{1-\delta})$, where the number of dense subintervals appearing inside a blue cluster_node is a function of n.

By setting $I_{max(1)} = \Theta(n^{1-\delta})$, we get the following:

$\Theta(n^{1-\delta}) = \Theta(n) - f(n) \cdot \Theta(n^{1-\delta})$. The latter means that: $f(n) = \frac{\Theta(n) - \Theta(n^{1-\delta})}{\Theta(n^{1-\delta})}$ or $f(n) = \Theta(n^{\delta})$. The property 1 follows:

Property 1. The number of consecutive dense subintervals is $f(n) = \Theta(n^{\delta})$.

So, now it's time to formally define what we mean with the term *any realistic distribution*.

Definition 1. Let $\mu(\cdot)$ be any random distribution in which there are sparse subintervals of $\Theta(n)$ sparseness and dense consecutive subintervals of $\Theta(n^{1-\delta})$

sparseness, where $0 < \delta < 1$. If the number of consecutive dense subintervals satisfies the property 1, then the $\mu(\cdot)$ random distribution is called *realistic distribution*.

We have to denote here that, the most known non-smooth (bad) distributions like skew, zipfian, powlaw, beta e.t.c. satisfy the property 1 for many real applications, thus would be called realistic for a huge variety of applications.

In other words, for any *realistic* distribution, each blue cluster_node in NEFOS structure satisfies the *smooth* property.

So, if the located cluster_node is red, then its load is polylogarithmic and the Lazy B-tree index requires a logarithmic number of I/Os:
$T_2(N) = O(\log_B(poly \log n))$ or $T_2(N) = O(\log_B \log n)$ I/Os or block-transfers.

If the located cluster_node is blue then its load may be $\Theta(n)$ in worst-case. Moreover, it's obvious that for any *realistic* distribution, each blue cluster_node satisfies the *smoothness* property. For this reason, we organize each blue cluster_node as an ISB-tree. In this case, $T_2(n)$ becomes as follows:

$$T_2(n) = O(\log_B \log \Theta(n)) \tag{2}$$

As a result, the total processing time requires $T(n) = T_1(n) + T_2(n)$ I/Os and the theorem follows:

Theorem 5. Exact-match queries in the NEFOS structure require $O(\log_B \log n)$ I/Os for any *realistic* input distribution.

Having located the target cluster_node for key k_ℓ and exploiting the order of keys in each bucket, range queries of the form $[k_\ell, k_r]$ require an $O(\log_B \log n + |A|/B)$ I/Os, where $|A|$ is the number of cluster_nodes between the buckets responsible for k_ℓ, k_r respectively that are accessed in a block manner. The theorem follows.

Theorem 6. Range queries of the form $[k_\ell, k_r]$ in the NEFOS structure require $O(\log_B \log n + |A|/B)$ I/Os for any *realistic* input distribution, where $|A|$ is the answer size.

Finally, provided that the position of update is given, meaning that we have already located the target cluster_node, it remains to insert/delete the key inside the cluster_node. Since, the latter is structured either as a Lazy-B tree or as an ISB-tree, the theorem 7 follows:

Theorem 7. Update queries in NEFOS require $O(1)$ I/Os for rebalancing operations in worst-case, provided that the update position is given.

4 Experimental Evaluation

In this section, we investigate the practical merits of the NEFOS structure. Our prime concern is to (merely) investigate the practical difference of the *asymptotic complexities* (in block transfers) of search and rebalancing operations between the NEFOS structure, the ISB-tree and a cache-aware B-tree. Although there are several cache-aware B-tree variants, all of them exhibit the same asymptotic

complexities in block transfers except for lazy B-tree which guarantees constant number of block transfers for update operations. Since lazy B-tree has been incorporated into ISB-tree in order to speedup the update operations, we compare the performance of NEFOS with the ISB-tree and a simple variant of the cache-aware B-tree. Moreover, we do not compare the performance of our rebalancing operations (after an update) with hashing schemes and their variants, since the expected-case analysis of such schemes usually assumes *uniform* input distributions (or input distributions that produce uniform hash key values), and hence they exhibit poor worst-case performance for update operations. In our experimental study, we have considered both synthetic and real-world data.

4.1 Synthetic Data

For evaluation purposes we used the Java NEFOS-simulator (source code of NEFOS index is available at http://www.ionio.gr/~sioutas/New-Software.htm). The NEFOS-simulator is extremely efficient delivering $> 100,000$ cluster nodes in a single computer system, using 32-bit JVM 1.6 and 1.5 GB RAM and full GUI support. When 64-bit JVM 1.6 and 5 RAM is utilized the NEFOS-simulator delivers $> 500,000$ cluster nodes and full GUI support in a single computer system. We have conducted an experimental study making the customary assumption that the page size is 4096 bytes, the length of each key is 8 bytes, and the length of each pointer is 4 bytes. Consequently, each block contains $B = 341$ elements. We considered data sets of size $n_0 \in [10^6, 10^{12}]$ elements generated by a variety of smooth distributions, namely uniform, regular, normal and Gaussian and non-smooth distributions, namely beta and pow-law. We compared the implementation of NEFOS, with the ISB-tree and that of a B-tree on the same data sets. Our main concern was to measure the performance, in simulated block transfers (I/Os), of the search and update operations.

The experimental results regarding the search operations are reported in Fig. 6 and 7. The sequence σ of search operations had length equal to its corresponding data set and the reported values are averages over the whole sequence. Our

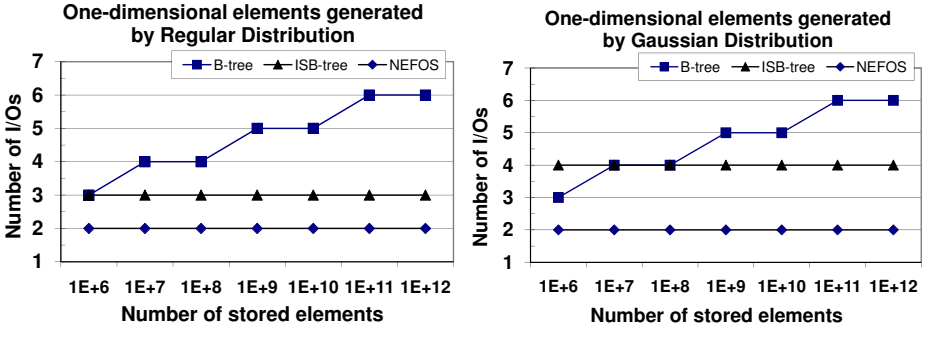


Fig. 6. Search performance for regular distributions (left) and Gaussian distributions (right)

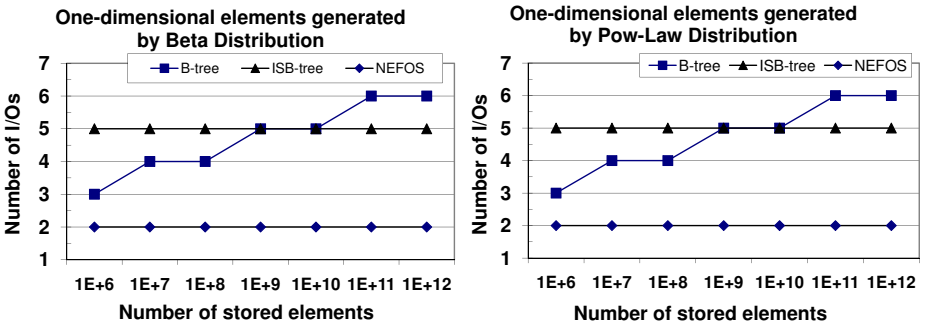


Fig. 7. Search performance for non-smooth beta (left) and powlaw distributions (right)

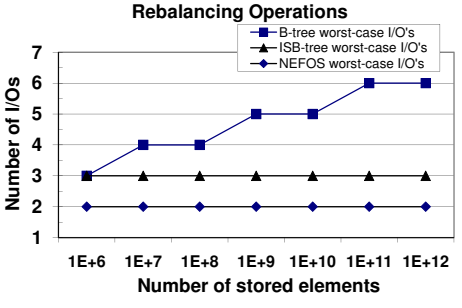


Fig. 8. Block transfers of rebalancing operations after an update

experiments revealed that the expected number of block transfers in NEFOS structure remains constant even for gigantic data sets (Terabytes - TB).

Regarding the number of block transfers required for rebalancing after an update operation to the data structure, we again considered the above values of $n_0 \in [10^6, 10^{12}]$ for our initial data sets upon which we performed update sequences of length $n_0/2$ and $2n_0$. The data structure is reconstructed every n_0 operations. Our experimental results are reported in Fig. 8. The values represent worst-case block transfers over the update sequence. We observe that the number of rebalancing operations in NEFOS structure is independent of the distribution.

4.2 Real-World Spatial Data

In this section, we deploy one-dimensional data taken from a real-world spatial dataset "LA rivers and railways" [Tiger1] and "LA streets" [Tiger2], containing 128971 and 131461 *M*inimum *B*ounded *R*ectangles (MBRs), respectively; see [23].

The one-dimensional data are taken by the x- and y-projections of MRBs and the values in each axis are normalized in [0,10000]. For all experiments, the disk page size is set to 512 bytes, the length of each key to 8 bytes, and the length of each pointer to 4 bytes. Consequently, each block contains $B = 42$ elements.

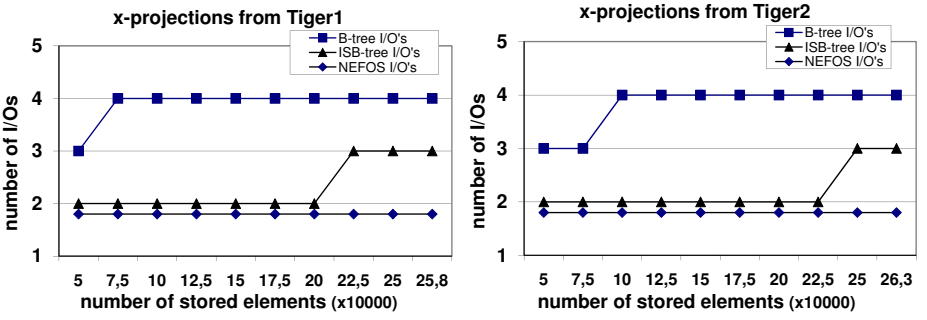


Fig. 9. Search performance for MBR's x-projections of [Tiger1] (left) and [Tiger2] (right)

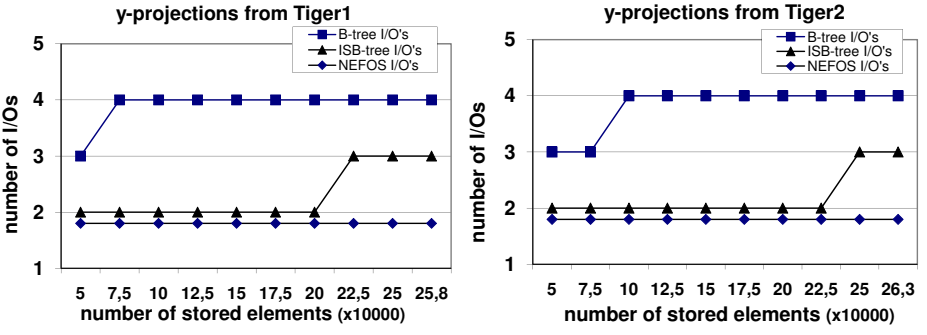


Fig. 10. Search performance for MBRs' y-projections of [Tiger1] (left) and [Tiger2] (right)

We use a relatively small page size so that the number of nodes in an index simulates realistic situations, where the data set cardinality is higher. A similar methodology was also used in [4].

Fig. 9 and Fig. 10 depict the efficiency of NEFOS structure on searching for real spatial one-dimensional data. In particular, in Fig. 9 we measured the number of I/Os required for search operations during the insertion of a total of $2 \times 128971 = 257942$ and of $2 \times 131461 = 262922$ x-projections from [Tiger1] and [Tiger2], respectively. Similarly, in Fig. 10 we measured the number of I/Os required for search operations during the insertion of a total of $2 \times 128971 = 257942$ and of $2 \times 131461 = 262922$ y-projections from [Tiger1] and [Tiger2], respectively.

Fig. 11 and Fig. 12 depict the efficiency of NEFOS structure on updating real spatial one-dimensional data. In Fig. 11 we measured the number of I/Os required for the rebalancing operations during insertions of a total of $2 \times 128971 = 257942$ x-projections and of $2 \times 131461 = 262922$ x-projections from [Tiger1] & [Tiger2], respectively. In the same way, in Fig. 12 we measured the number of I/Os required for rebalancing operations during insertions of $2 \times 128971 = 257942$

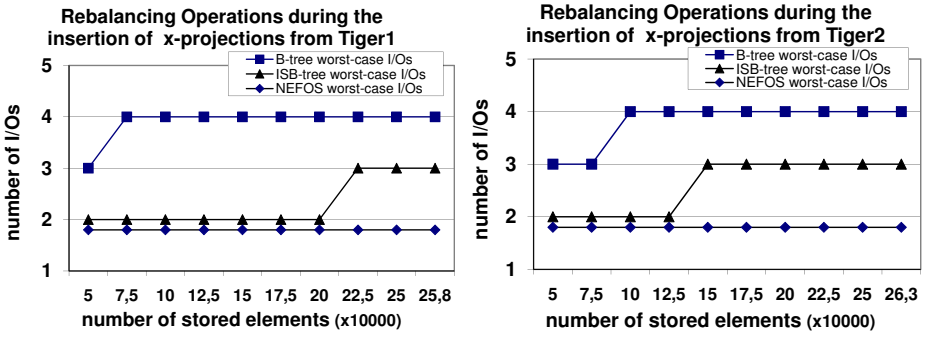


Fig. 11. Performance of rebalancing operations after an update for MBRs' x-projections of [Tiger1] (left) and [Tiger2] (right)

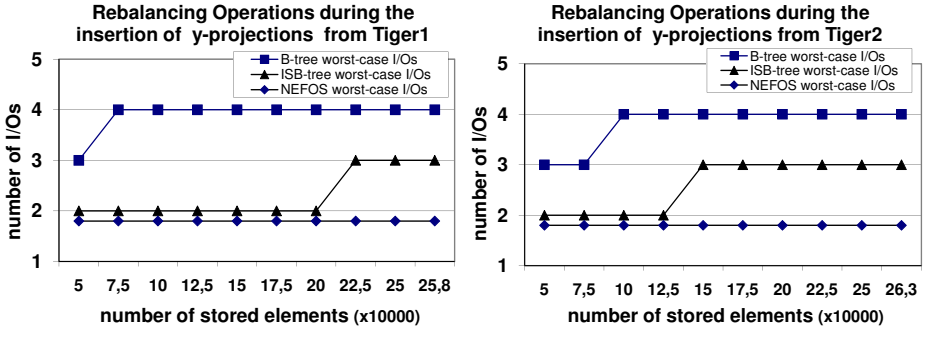


Fig. 12. Performance of rebalancing operations after an update for MBRs' y-projections of [Tiger1] (left) and [Tiger2] (right)

y-projections and of $2 \times 131461 = 262922$ y-projections from [Tiger1] & [Tiger2], respectively.

The above experiments show that NEFOS has approximately the same behaviour with ISB-tree requiring no more than 2 I/Os on average for both searching and rebalancing operations. This stems from the fact that the MBRs' projections from the data sets [Tiger1] & [Tiger2] follow an almost uniform distribution, due to the almost uniform decomposition of spatial maps. Better performance in [Tiger 2] is due to the fact that this is a dense spatial map and hence the derived one-dimensional data produce densely populated elements.

As a final remark, we note that there are applications with uniform key sizes larger than 8 bytes, resulting in a smaller value of B. The main example of such applications involve manipulation of strings. In this case, the size of the block may be as small as 2. Consequently, in such cases the NEFOS structure exhibits a much better performance.

5 Conclusions

We presented NEFOS (NEsted FOrest of balanced treeS), the first cache-aware indexing scheme, which supports expected w.h.p. sub-logarithmic range query processing for any (unknown) realistic input distribution. Moreover, NEFOS is the first concrete access method, which works for both I/O and RAM model, avoiding any kind of transformation or adaptation. The innovation of our solution was also verified by an accompanying experimental study. We leave for journal version a more detailed theoretical analysis as well as an exhaustive experimental evaluation of multi-dimensional exact-match and range queries.

References

1. Arge, L., de Berg, M., Haverkort, H.J., Yi, K.: The Priority R-Tree: A Practically Efficient and Worst-Case Optimal R-Tree. In: SIGMOD Conf., pp. 347–358 (2004)
2. Aggarwal, A., Vitter, J.S.: The Input/Output Complexity of Sorting and Related Problems. C. ACM 31(9), 1116–1127 (1988)
3. Andersson, A., Mattson, C.: Dynamic Interpolation Search in $o(\log \log n)$ Time. In: Lingas, A., Carlsson, S., Karlsson, R. (eds.) ICALP 1993. LNCS, vol. 700, pp. 15–27. Springer, Heidelberg (1993)
4. Beckmann, N., Krigel, H., Schneider, R., Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: SIGMOD (1990)
5. Bayer, R., McCreight, E.: Organization of large ordered indexes. Acta Informatica 1, 173–189 (1972)
6. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. C. ACM 51, 107–113 (2008)
7. Dietz, P., Raman, R.: A constant update time finger search tree. Information Processing Letters 52, 147–154 (1994)
8. Fagin, R., Nievergelt, J., Pippinger, N., Strong, H.R.: Extendible Hashing-A fast access method for dynamic files. ACM Trans. Database Systems 4(3), 315–344 (1979)
9. Ferragina, P., Grossi, R.: The String B-tree: A New Data Structure for String Search in External Memory and Its Applications. Journal of the ACM 46(2), 236–280 (1999)
10. Fox, E., Chen, Q., Daoud, A.: Practical Minimal Perfect Hash Functions for Large Databases. C. ACM 35(5), 105–121 (1992)
11. Kaporis, A., Makris, C., Sioutas, S., Tsakalidis, A., Tsichlas, K., Zaroliagis, C.: Improved Bounds for Finger Search on a RAM. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 325–336. Springer, Heidelberg (2003)
12. Kaporis, A., Makris, C., Mavritsakis, G., Sioutas, S., Tsakalidis, A., Tsichlas, K., Zaroliagis, C.: ISB-Tree: A New Indexing Scheme with Efficient Expected Behaviour. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 318–327. Springer, Heidelberg (2005)
13. Knuth, D.E.: Deletions that preserve randomness. IEEE Trans. Softw. Eng. 3, 351–359 (1977)
14. Lehman, P., Bing Yao, S.: Efficient Locking for Concurrent Operations on B-Trees. ACM Trans. Database Systems 6(4), 650–670 (1981)
15. Levcopoulos, C., Overmars, M.H.: Balanced Search Tree with $O(1)$ Worst-case Update Time. Acta Informatica 26, 269–277 (1988)

16. Litwin, W.: Linear Hashing: A new tool for files and tables addressing. In: International Conference on Very Large Databases, vol. 6, pp. 212–223 (1980)
17. Litwin, W., Lomet, D.: A New Method for Fast Data Searches with Keys. IEEE Software 4(2), 16–24 (1987)
18. Manolopoulos, Y., Theodoridis, Y., Tsotras, V.: Advanced Database Indexing. Kluwer Academic Publishers, Dordrecht (2000)
19. Mehlhorn, K., Tsakalidis, A.: Dynamic Interpolation Search. Journal of the ACM 40(3), 621–634 (1993)
20. Raman, R.: Eliminating Amortization: On Data Structures with Guaranteed Response Time. PhD Thesis, Dept. of Computer Science, University of Rochester, New York; Technical Report TR-439 (1992)
21. Seeger, B., Larson, P.A.: Multi-Disk B-trees. In: Proc. SIGMOD Conference, pp. 436–445 (1991)
22. Srinivasan, V., Carey, M.J.: Performance of B+ Tree Concurrency Algorithms. VLDB Journal 2(4), 361–406 (1993)
23. Theodoridis, Y.: The R-tree Portal (2003), http://www.rtreeportal.org, [Tiger1] and [Tiger2] data sets in http://www.rtreeportal.org
24. Vitter, J.S.: External memory algorithms and data structures: dealing with massive data. ACM Computing Surveys 33(2), 209–271 (2001)
25. Vitter, J.S., Shriver, E.A.M.: Optimal Algorithms for Parallel Memory I: Two-Level Memories. Algorithmica 12(2-3), 110–147 (1994)
26. Willard, D.E.: Searching Unindexed and Nonuniformly Generated Files in log log N Time. SIAM Journal of Computing 14(4), 1013–1029 (1985)
27. Willard, D.E.: Examining Computational Geometry, van Emde Boas Trees, and Hashing from the Perspective of the Fusion Tree. SIAM Journal of Computing 29(3), 1030–1049 (2000)

Reuse-Oriented Mapping Discovery for Meta-querier Customization

Xiao Li and Randy Chow

Department of CISE at University of Florida
{xl1,chow}@cise.ufl.edu

Abstract. With the tremendous growth in (semi-)structured information, in particular, the databases behind the deep Web, customized integration of data sources is highly desirable to accommodate various user needs. To build and maintain a potentially large number of customized integration systems (called meta-queriers), the first important step is to discover the mappings among their query forms for enabling interoperability between the meta-queriers and user-selected data sources. This paper proposes a reuse-oriented solution to mapping discovery by exploiting existing mappings. For facilitating mapping reuse, ontology-based and changed-oriented models are introduced to abstract mappings respectively from the mapping peers (the mappings in the same domain) and the mapping evolution (the mappings from the previous versions). A human-friendly validation strategy is proposed in the pursuit of wide and active participation of non-technical volunteers. Our experimental results on real-world data sets confirm the feasibility and effectiveness of this solution.

Keywords: customization, deep web, schema matching, ontology, data integration.

1 Introduction

Deep Web contains an increasing number of (semi-)structured data sources that are mostly invisible to traditional search engines. Through integrating the query forms (a.k.a, local forms) of the data sources, meta-queriers enable users to query them simultaneously by entering search criteria in a uniform query form (a.k.a, global form). To reformulate the user queries over the global form to the queries in terms of the local forms, the meta-queriers rely on the mappings between the global and local forms.

Due to the ever-increasing number of available data sources and sophistication of users, it is highly desirable (and in many cases necessary) to allow for customization of meta-queriers based on users preferences (or necessity) [29], even in the same application domain. In this paper, we allow users to customize the construction of application-specific meta-queriers by selecting their preferred data sources. In this context, a large number of customized meta-queriers need to be constructed. When the scale of meta-queriers to be built becomes large due to various user needs, it is impractical to apply the traditional approaches (e.g., WISE-Integrator [14] and Meta-Queriers [5]), which aim at automating the construction of a single meta-querier.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 78–93, 2011.

In addition, both data sources and system users are self-governing agents that are autonomous from meta-queriers. Autonomy of data sources means that these sources normally update their systems without any notification. In case of the changes in their local forms, their inclusive data are normally unavailable/invisible to the pre-configured meta-queriers. User needs and preferences are also not static. For example, users might want to insert new data sources into meta-queriers for retrieving more relevant data. Thus, meta-queriers need to be maintained for adapting to changing user requirements.

In construction and maintenance of meta-queriers, the first important step is to discover the mappings between global forms and local forms, which play a critical role in query reformulation. To attack meta-querier customization, the challenge is discovery of a potentially large number of mappings. However, discovery of an individual mapping, especially with a complex many-to-many expression, is well known as an AI-complete problem [12] [19]. In our solution approach, we turn the scalability challenge into the reuse potential of numerous existing mappings. Instead of directly reusing the individual mappings as in the previous studies [22] [26] [10] [7], our strategy is to use the abstraction from the mappings and their evolution for better performance. Overall, our reuse-oriented solution makes the following contributions:

• *Modeling of existing mappings.* Efficient reuse of existing mappings is based on abstraction of the mappings. We propose two models for understanding the mappings and recording their evolution. Based on our *ontology-based* model, *M-Ontology* enables unordered mappings in the same domain to form a well-organized ontology. Following our *change-oriented* model, *MO-Repository* can identify and record the evolution processes by using bipartite graphs. Both models enhance reuse potential from the existing mappings.

• *Discovery of complex mappings.* To discover new mappings, our reuse-oriented discovery algorithm employs their evolution processes (the previous versions) and peers (the other mappings in the same domain). This approach enables discovery of many-to-many complex mappings by finding one-to-one correspondences (i.e., one-to-one mappings without the expressions). Most algorithms in the prior studies [11] [22] can be easily integrated into our solution, since discovery of one-to-one correspondence is one of the major goals in schema matching.

• *Validation of numerous mappings.* It is difficult for a small group of domain experts to verify and correct a potentially large number of mappings. A *mass collaboration* strategy is proposed to distribute the workload among community members, which could include both ordinary users and domain experts. For participation of non-technical volunteers, our solution is intentionally straightforward to ordinary users. That is, it is easier for them to understand the process, validate the results, and improve the performance.

The rest of this paper is organized as follows. In Section 2, we present two mapping models with the corresponding repositories. Based on the models, Section 3 introduces a reuse-oriented algorithm for mapping discovery. We discuss the experimental results in Section 4 and related work in Section 5. Section 6 concludes the paper with potential directions for future work.

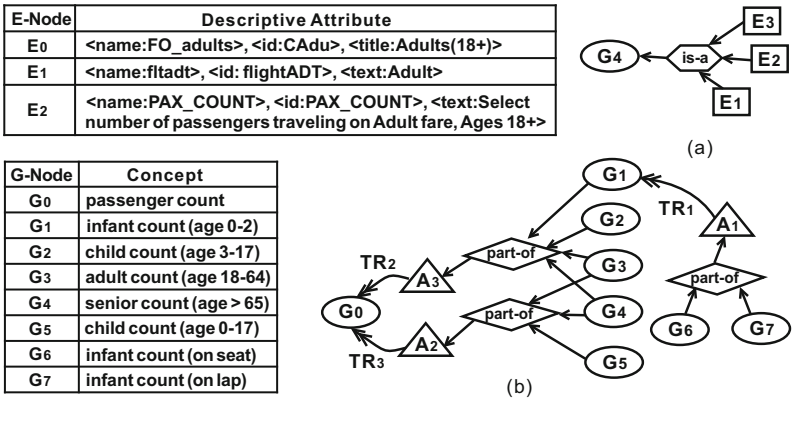
2 Modeling of Mappings

The mappings among query forms can be divided into multiple independent mappings among their inclusive query conditions [28] [16] (also referred to as *schema elements*). Since this paper only focuses on HTML query forms [1], each schema element consists of a HTML control (e.g., a checkbox), its associated descriptive attributes (e.g., id, name and values) and instances (i.e., possible user inputs). In the context of meta-querier customization, the scale of such mappings could be considerably large since a potentially large number of meta-queriers need to be built to meet various user needs. However, the potential for discovering new mappings through reusing the previous ones can also be significant. Before discussing the details of our reuse-oriented mapping discovery algorithm, this section first presents the definition of mappings in meta-queriers, and then two models of mappings from the perspectives of their semantics and evolution processes.

Definition 1. *An element mapping* map_T^S *(also called a mapping instance) is an instance of a specific relation from a query form* QF_S *to* QF_T*. It can be represented by a tuple* $\langle ElistS, ElistT, Exp_T^S \rangle$*, where* $ElistS$ *and* $ElistT$ *are schema element lists respectively from* QF_S *and* QF_T*,* Exp_T^S *is a high-level declarative expression that specifies the transformation rules from* $ElistS$ *to* $ElistT$*. An element mapping without* Exp_T^S *is called as a correspondence* $corr_T^S$*.*

2.1 Ontology-Based Mapping Modeling

From the viewpoint of mapping semantics, a mapping can be viewed as an instance of a relation connecting two concepts. The concepts can be automatically extracted from the schema elements associated with the mappings. The relations are higher-level abstraction from mappings. Following this perspective, mappings can be modeled by using a domain-specific task ontology (called *M-Ontology*) for expressive power and content coverage.



E-Node	Descriptive Attribute
E0	<name:FO_adults>, <id:CAdu>, <title:Adults(18+)>
E1	<name:fltadt>, <id: flightADT>, <text:Adult>
E2	<name:PAX_COUNT>, <id:PAX_COUNT>, <text:Select number of passengers traveling on Adult fare, Ages 18+>

G-Node	Concept
G0	passenger count
G1	infant count (age 0-2)
G2	child count (age 3-17)
G3	adult count (age 18-64)
G4	senior count (age > 65)
G5	child count (age 0-17)
G6	infant count (on seat)
G7	infant count (on lap)

Fig. 1. A fragment of M-Ontology for air-ticket booking

M-Ontology can be represented as a directed acyclic graph with three types of nodes, as shown in Fig. 1: E-Nodes (squares), G-Nodes (circles) and A-Nodes (triangles). Each node denotes a concept with a set of associated instances. The edge corresponds to a relation in data transformation that is indicated in a specific mapping instance. (1) E-Nodes are the most fundamental concept units. Each E-Node corresponds to a schema element. For example, E_1 in Fig. 1 represents a schema element about "adult count". (2) G-Nodes are the generalization of E-Nodes that share the same semantics and instance formats but their syntactic representations are different. For example, the concept of G_4 is generalized from three verified E-Nodes. Although these E-Nodes have equivalent semantics and instances format, their descriptive attributes are different in numbers, types and values. The generalization process is done through the incremental clustering of schema elements via our proposed representative object-based clustering algorithm. (3) An A-Node is generated by aggregating an (ordered/unordered) list of G-Nodes. For example, two A-Nodes A_2 and A_3 represent the concept of "passenger count" by aggregating multiple G-Nodes. A_2 has a coarse-grained classification of the concept "passenger count", using "child count (age 0-17)"(G_5) to represent "infant count (age 0-2)"(G_1) and "child count (age 3-17)"(G_2). The composition of concepts can be obtained through insertion of many-to-many mappings. (4) A T-Edge connecting two nodes represents a transformation relation. For example, a T-Edge TR_1 is created for transforming instances from A_3 to G_0, where they respectively use different formats to represent the same concept "passenger count". These transformation relations can be directly obtained from the mapping expression Exp_T^S.

In the prior research, we develop a basic ontology-based mapping management scheme (i.e., M-Ontology) [17] for meta-querier customization [18]. Mappings are managed under a win-win strategy: users are responsible for creating, validating and correcting the results from mapping discovery made either by the machines or the other users. Validated mappings are automatically recycled in our mapping insertion algorithm for building the M-Ontology incrementally. In turn, the validated M-Ontology contents (i.e., concepts and relations) can also be employed for discovering new mappings in an automated manner.

2.2 Change-Oriented Mapping Modeling

Mappings are not static but dynamic in the context of meta-querier customization, where both end users and data sources are autonomous agents that are independent from meta-queriers. Scenarios of mapping evolution can be characterized into two types:

1) *External changes*: The changes in the schema elements of either local or global forms entail the updates to the related mappings, as shown in Fig. 2 (a) and (b). Changes in local forms are made in an untraceable manner. The major changes are observed in terms of the functionalities and representation of data sources. For example, if a car-rental local form wants to support new car models (e.g., 2011 Ford Fiesta), the corresponding entries need to be included in its car-model control (e.g., a selection menu). Furthermore, changes in global forms

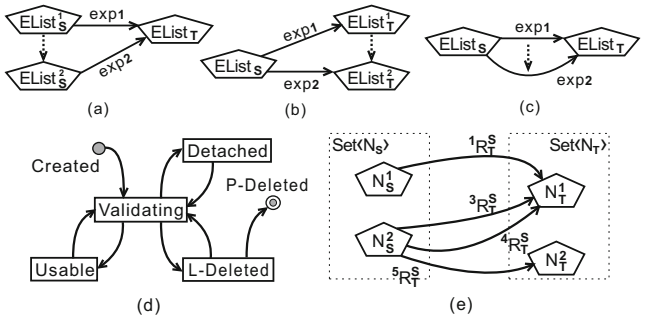


Fig. 2. Mapping evolution scenarios and a mapping object example

occur due to the evolution of its underlying local forms and user-specified source selection. These changes are traceable.

2) *Internal changes*: Modification of mappings comprises changes on the mapping composition and updates of their related context information, i.e., mapping metadata. A mapping instance can be created from scratch, or from another instance with some modification. There is no guarantee that these mapping instances discovered by machines or humans are completely free from errors and always function well. To correct the errors, the potential modification includes changes on the expressions (i.e., Exp_T^S) and the element lists (i.e., $EList_S$ and $EList_T$), as shown in Fig. 2 (a), (b) and (c),

Based on the above observations, we design a change-oriented mapping model to preserve mapping evolution.

Mapping Lifecycle: Each mapping instance has its own lifecycle starting from the initial creation to the physical deletion. The state transitions are determined by its validation status. Fig. 2(d) illustrates a state diagram with the four states that a mapping can be in: *Validating, Usable, Detached*, and *L-Deleted*. The *Validating* state of a mapping indicates that its current correctness is undetermined. It remains in this state until humans or machines validate it completely. A newly created mapping begins its lifecycle in the *Validating* state. The state of an existing mapping transitions into the *Validating* when its correctness status is changed. While a mapping is at the *Usable* state (i.e., ready for being utilized), it is able to perform correctly in the associated meta-queriers. When a previously correct mapping is identified to be incorrect, it enters the *Detached* state. Such a mapping might be useful for discovering new mappings. For the mappings that are invaluable (e.g., never correct), they enter *L-Deleted* state (i.e., logically deleted). The lifecycle of a mapping is finished when it is physically deleted (*P-Deleted*).

Mapping Objects: Between one query form QF_S and another QF_T, there exist a set of *mapping objects*. A mapping object $mapObj_T^S$ denotes a specific relation between QF_S and QF_T (i.e., $QF_S \rightarrow QF_T$ and $QF_T \rightarrow QF_S$). It can be represented by a bipartite graph whose edges $Set\langle R_T^S, R_S^T \rangle$ only connect the nodes from two disjoint node sets $Set\langle N_S \rangle$ and $Set\langle N_T \rangle$. Each node in $Set\langle N_S \rangle$ and $Set\langle N_T \rangle$

respectively corresponds to an element list from QF_S and QF_T. Each solid edge represents a mapping instance of $mapObj_T^S$. Based on the semantics, every pair of the nodes respectively in $Set\langle N_S\rangle$ and $Set\langle N_T\rangle$ is a correspondence $corr_T^S$.We assume that all the mapping instances in the same direction (e.g., $QF_S \rightarrow QF_T$) are independent of each other. This assumption is feasible since two instances can be easily merged until there does not exist any constraint between them.

Each mapping object consists of a group of mapping instances that represent the same relation between two query forms. These instances are regarded as different versions of this object. Thus, a mapping object can be represented using a bipartite graph $\langle Set\langle N_S\rangle, Set\langle N_T\rangle, Set\langle R_T^S/R_S^T\rangle\rangle$, as illustrated in Fig. 2(e). For example, assuming the original mapping object is only a single mapping instance ${}^1R_T^S$, another instance ${}^2R_T^S$ with the inverse direction is added into the object. When the external changes occur in QF_S, two original mapping instances, ${}^1R_T^S$ and ${}^2R_T^S$, are detached and a new mapping instance ${}^3R_T^S$ is automatically discovered by the machine. Then, since ${}^3R_T^S$ is incorrect based on manual validation, it is logically deleted and replaced by another mapping instance ${}^4R_T^S$. After the internal changes in QF_S, all the existing usable mappings are detached. Finally, ${}^5R_T^S$ is the only usable mapping instance.

Complementary to M-Ontology, MO-Repository is a repository of mapping evolution. It records the evolution of mappings that serve the meta-queriers in a specific application domain. That is, it stores the corresponding mapping objects $Set\langle MapObj\rangle$, each of which $mapObj_T^S$ denotes a concrete relation between two query forms QF_S and QF_T. $mapObj_T^S$ consists of the different versions of such a relation. Each version corresponds to an individual mapping instance. The following section explains how to use M-Ontology and MO-Repository for discovering new mappings.

3 Reuse-Oriented Mapping Discovery

Mapping discovery is a critical operation for meta-querier construction and maintenance. However, existing techniques [11] [22] are not adequate in discovering the mapping automatically, especially those with non-equivalence expressions. Our proposed solution aims at decreasing the complexity and workloads of mapping discovery made by humans. Thus, between two query forms QF_S and QF_T, our algorithm outputs not only the active mapping instances map_T^S but also their previous versions (i.e., the mapping object $mapObj_T^S$). If the mapping expressions cannot be found, the correspondences $corr_T^S$ are also offered for facilitating manual mapping discovery.

With a domain-specific M-Ontology MO and MO-Repository MOR, the reuse-oriented algorithm is to discover $\mathrm{Set}\langle map_T^S, corr_T^S, mapObj_T^S\rangle$ from QF_S to QF_T, as illustrated in Fig.3. We first extract query conditions from query forms QF_S and QF_T to respectively generate source and target element sets $Set\langle e_S\rangle$ and $Set\langle e_T\rangle$. Since automatic extraction is not the focus of this paper (see [28] [15] [16] for more information), the current implementation is based on manual extraction. The overall algorithm can be split into three phases: 1) mapping discovery through MOR; 2) mapping discovery through MO; 3) human validation and correction of the discovered mappings. The details are discussed below.

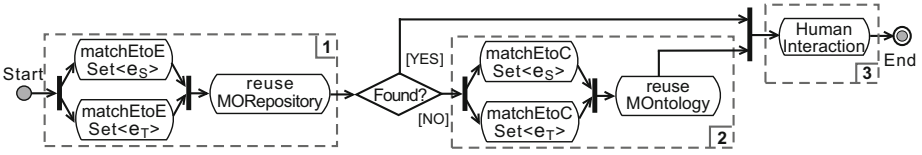


Fig. 3. Reuse-oriented algorithm for mapping discovery

3.1 Mapping Reuse through MO-Repository

In the first phase, discovery through MO-Repository MOR is a search process for reusing the existing mapping instances between the same pair of query forms. To match the query forms from QF_S to QF_T, we first seek two sets of one-to-one element correspondences $Set\langle(e_S, e_{MOR})\rangle$ and $Set\langle(e_T, e_{MOR})\rangle$, where e_{MOR} is the elements from MOR. Through these correspondences, the eligible mapping instances map_T^S, correspondences $corr_T^S$ and mapping object $mapObj_T^S$ are selected by reasoning on MOR. The elements in $Set\langle e_S\rangle$ and $Set\langle e_T\rangle$ are sent to the second procedure with the element correspondences $Set\langle(e_S, e_{MOR})\rangle$ and $Set\langle(e_T, e_{MOR})\rangle$. The core is the implementation of two functions: matchEtoE and reuseMORepository.

• matchEtoE selects and returns a set of element correspondences $Set\langle(e_X, e_{MOR})\rangle$ for each side (that is, X can be S or T). The objective of correspondence selection is to maximize the utility value as computed by,

$$\sum_{e_X \in X} \sum_{e_{MOR} \in MOR} R(e_X, e_{MOR}) \times a(e_X, e_{MOR}),$$

subject to the following constraints:

$$\sum_{e_X \in X} a(e_X, e_{MOR}) \leq 1, \qquad \sum_{e_{MOR} \in MOR} a(e_X, e_{MOR}) \leq 1, \quad a(e_X, e_{MOR}) \in \{0, 1\}$$

and $R(e_X, e_{MOR}) \in \{0, 1\}$ for $e_X \in X$, $e_{MOR} \in MOR$,

where, all the elements e_X should be from the same query form X (i.e., QF_S or QF_T), and element e_{MOR} is from MO-Repository MOR and with the same query form. $a(e_X, e_{MOR})$ takes the value 1 if a correspondence between e_X and e_{MOR} is selected, and 0 otherwise. The feasible result must guarantee that each e_{MOR} is assigned to at most one e_X and each e_X is assigned to at most one e_{MOR}. The similarity value $R(e_X, e_{MOR})$ is equal to 1 only when the similarity value between e_X and e_{MOR} is larger than a pre-defined threshold δ; otherwise it is assigned with a value 0. Their similarity values can be calculated by the following formula:

$$w_1 \times sim_{DA-DA}(e_X, e_{MOR}) + w_2 \times sim_{INS-INS}(e_X, e_{MOR})$$

where w_1 and w_2 are two weights in [0,1] such that $\sum_{k=1}^{2} w_k = 1$. The similarity values returned from sim_{DA-DA} and $sim_{INS-INS}$ show the extent how similar element e_X and element e_{MOR} are regarding to their Descriptive Attributes and INStances, respectively. The element attributes and instances are first normalized by standard NLP (natural language processing) techniques: tokenization, stop-word removal and stemming. The resulting texts are then considered as

bags of words, that is, unordered sets of words. To compare the similarities of these word sets, a hybrid matcher is employed by combining several typical linguistics-based matchers, such as WordNet-synonyms distances, 3-gram distances and Jaccard distances. Many algorithms for schema/ontology matching have been proposed (see the surveys[22] [11]). Most of them can be easily introduced in this phase. As this is not the major target of this paper, the details are not further elaborated here.

The decision of these element correspondences is a generalized assignment problem, which is a NP-hard problem. However, a simple greedy solution can quickly attain the results in this context. First, the similarity matrix size and number is not big [4]. For example, normally, the query forms in flight-ticket searching contain around 10 functional components. The query forms are compared with only the query form with the same query form identifier. Second, the similarity matrix is sparse. Very few similarity values are equal to non-zero.

• reuseMORepository is to determine mapping instances, correspondences and mapping objects (i.e., $\text{Set}\langle map_T^S, corr_T^S, mapObj_T^S\rangle$) through the identified element correspondences. 1) A mapping object is selected only if the corresponding bipartite graph contains at least one node in $Set\langle N_S\rangle$ or $Set\langle N_T\rangle$ that is *fully hooked*, i.e., all the elements of this node appear in element correspondences $Set\langle(e_S, e_{MOR})\rangle$ or $Set\langle(e_T, e_{MOR})\rangle$. For example, in Fig. 2(e), if all the elements in N_S^1 are fully hooked, the mapping object will be included in the return set. 2) $corr_T^S$ is determined based on a property of MO-Repository. The elements in every pair of nodes respectively from $Set\langle N_S\rangle$ and $Set\langle N_T\rangle$ is a correspondence. For example, in Fig. 2(e), the elements in N_S^1 and N_T^2 constitute a correspondence. Thus, if both two node sets $Set\langle N_S\rangle$ and $Set\langle N_T\rangle$ have at least one node that are fully hooked, the elements $e_{S/T}$ that are linked to these node pairs constitute new correspondences. 3) The mapping instance is discovered through examining each identified correspondence: i) whether there exists an edge to connect the nodes in the corresponding bipartite graph; ii) whether the mapping state of this edge is *Detached* or *Usable*; iii) whether the edge direction is $QF_S \rightarrow QF_T$. If all the states are true, the correspondence with this edge can be regarded as a mapping instance map_T^S. Finally, $Set\langle map_T^S, corr_T^S, mapObj_T^S\rangle$ are found via reuse of the previous mappings in MOR.

3.2 Mapping Reuse through M-Ontology

The second phase is to discover the mappings through exploiting the conceptual abstraction from the existing mappings in the same domain. To determine the mappings from QF_S to QF_T, all the undetermined schema elements in $Set\langle e_S\rangle$ and $Set\langle e_T\rangle$ are first classified into the appropriate G/A-Nodes in M-Ontology MO. Based on classification results, we can discover new mappings by reusing the mappings associated with the corresponding G/A-Nodes. The implementation is through two major functions: matchEtoC and reuseMOntology.

• matchEtoC is to classify each undetermined element $e_{S/T}$ in $Set\langle e_S\rangle$ and $Set\langle e_T\rangle$ to the appropriate G/A-Nodes from the M-Ontology MO. The resulting G/A-Nodes constitute two concept sets $ASet_S$ and $ASet_T$. A-Nodes are selected if all the inclusive G-Nodes are already identified. Thus, the major focus is how

to find an appropriate G-Node gn for a given element $e_{S/T}$. We design a two-step solution that combines both *history-based* and *semantics-based* selection strategies. The first-step selection is based on mapping evolution. We obtain the element correspondences $Set\langle(e_S, e_{MOR})\rangle$ or $Set\langle(e_T, e_{MOR})\rangle$ from matchEtoE in the first phase (discussed in Section 3.1). If the correspondences are accurate, the G-Nodes that contain e_{MOR} should have the same semantics as $e_{S/T}$. Thus, all the eligible G-Nodes are chosen. The second step is based on the semantic similarity between undetermined elements $e_{S/T}$ and the representative objects ro of the G-Nodes. ro is represented by a tuple $\langle DA, DL, INS, IT\rangle$, where DA is a bag of descriptive words that can be generated through normalizing the descriptive attributes of all human-verified E-Nodes in gn using the aforementioned NLP techniques; DL is a set of descriptive labels that consist of the terms with the top-k term frequency from DA; INS and IT respectively represent the instances and their types obtained from the inclusive verified E-Nodes. When their types are identical, the similarity values can be calculated by the following formula:

$$w_1 \times sim_{DA-DA}(e_{S/T}, ro) + w_2 \times sim_{DA-DL}(e_{S/T}, ro) + w_3 \times sim_{INS-INS}(e_{S/T}, ro),$$

where w_1, w_2 and w_3 are three weights in [0,1] such that $\sum_{k=1}^{3} w_k = 1$. Due to the page limits, the algorithms are not explained in details since the idea is highly similar to the the algorithms mentioned in matchEtoE.

• reuseMOntology discovers $\text{Set}\langle map_T^S, corr_T^S\rangle$ (i.e., the mapping instances and correspondences) from M-Ontology. First, we identify two G/A-Node sets $RSet_S$ and $CSet_S$ whose nodes are reachable from the nodes in the concept set $ASet_S$. $RSet_S$ consists of the nodes that can be reached through a single T-Edge from any node in $ASet_S$. $CSet_S$ is the node union of maximal connected subgraphs of the nodes in $ASet_S$ (T-Edge direction is ignored here). Second, the desired concept-level mappings are the overlap $ASet_S \cap ASet_T$ and $RSet_S \cap ASet_T$. The first overlap denotes the semantics-equivalence mappings between two concepts. The second one indicates the concept mappings whose expressions are in the connected T-Edges. We also can deduce the target concept correspondences from the overlap $CSet_S \cap ASet_T$, although M-Ontology does not have a T-Edge to link them together. Finally, we can easily discover $\text{Set}\langle map_T^S, corr_T^S\rangle$ through the element classification (from matchEtoC) with these concept mappings and correspondences.

3.3 Validating and Correcting Mappings

The final phase is to verify and correct the machine-discovered mappings by human beings. Supporting meta-querier customization is likely to result in a large number of mappings to be discovered. It is impractical for a small set of domain experts and system designers to validate and correct these mappings. We propose the following strategies to decrease their workloads:

Mass collaboration. Although the to-be-discovered mappings might be numerous, meta-querier customization will also attract more users. We believe these users not only have enough motivation to be involved in the validation, but also are willing to volunteer to assist others. In our design, these users along with a small number of domain experts can practically form a collaborative

domain-specific community for meta-querier construction. We divide the error-prone procedure into three subsequent procedures with different difficulty levels. The tasks in the easiest level include two verification jobs: verifying if the classified schema elements are assigned to the correct G-Nodes and checking if the discovered T-Edges are appropriate to represent the relations among the schema elements. Normally, these basic tasks can be assigned to novices. In the second difficulty level, the tasks are determining unfound mappings and searching M-Ontology to seek correct G-Nodes, if existing, for unclassified elements. The experienced users are able to handle these jobs. At the most difficult level of tasks, M-Ontology needs to be updated (e.g., creating a new G-Node) for supporting new mapping instances. The maintenance of M-Ontology should be one of the major jobs of the domain experts.

Human friendliness. For any community-based system to be successful, it is absolutely necessary that the system must be easy to control. In our proposed discovery algorithms, we can incorporate the following special considerations in the design. First, in the function matchEtoE, correspondence selection is viewed as a utility maximization problem on similarity matrix $R(e_X, e_{MOR})$. Thus, the results can be improved through manual or automatic changes on $R(e_X, e_{MOR})$. For example, verification on mappings might trigger changes on the corresponding values. Second, in the function matchEtoC, manual modification on the representative objects of concept nodes (such as D-Labels and Instances) can improve the precision and recall. Third, graph-oriented interface is an essential requirement for systems with a large community of novice users. Thus, both mapping objects and mapping ontology can be represented by graphs. The related operations can be implemented with more straightforward graph operations using emerging HCI technologies (e.g., [6] [20]). Last but not least, even if the current contents of M-Ontology are correct and non-redundant, domain experts are also encouraged to involve in the enrichment of M-Ontology for improving the performance. The comments on nodes/edges are welcomed from the creators/modifiers. The information stored in metadata (such as annotation) is critical for understanding by others.

4 Experiments

We evaluate our reuse-oriented mapping discovery by conducting several sets of experiments on real-world query forms. The goal of experiments is to examine the feasibility and effectiveness of our approach. Specifically, the experiments are designed to answer the following three research questions in the context meta-querier customization: 1) Is it practical to find the target mappings/correspondences from M-Ontology? 2) Is our algorithm effective in identifying the mappings from M-Ontology? 3) Is our algorithm able to improve the effectiveness of mapping discovery with the incorporation of mapping evolution?

Data sets: We first collected 38 query-form URLs from the air-ticket booking data set in the UIUC web integration repository[2] after removing the inactive forms in May 2009. These query forms are manually extracted from their HTML codes and represented in Ontology Language (OWL). These OWL files

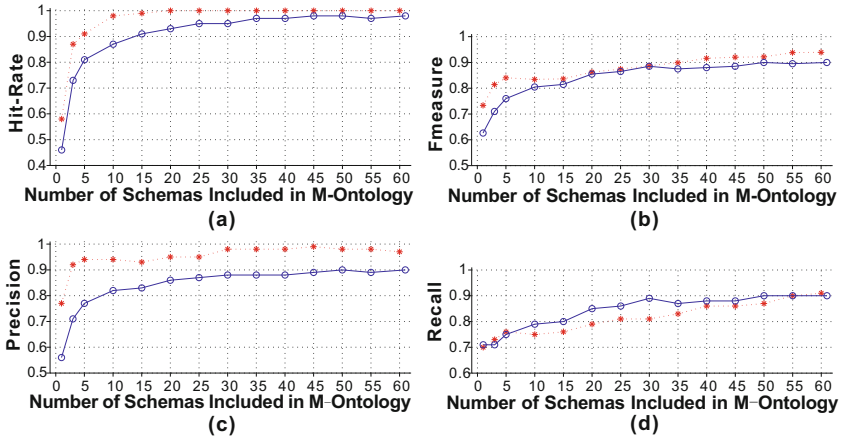


Fig. 4. Experiment results of mapping discovery through M-Ontology

form the first data set AIRO. After 22 months, we revisited the web pages containing these query forms, 23 of which were syntactically changed, and their corresponding OWL files are generated for them. The total 61 query forms constitute the second data set AIR. From these forms, 730 schema elements are manually classified based on their semantics. Each schema element is classified to a single G-Node and a single G/A-Subgraph. These classification results are used in the performance evaluation of the following four sets of experiments.

Experiment setup: In the following experiments, we assume that m Web query forms are already contained in M-Ontology/MO-Repository. For M-Ontology, the representative objects are automatically generated from the verified schema elements, without manual correction on the D-Attributes, D-Labels and Instances. We also assume that T-Edges have been created to connect the semantics-equivalent concept nodes. In the following experiment results, each value is obtained by an average of 100 samples. To ensure fairness and accuracy in measurement, each sample is randomly generated from the data sets without any duplicate. Four measures are used in each experiment: *Hit-rate* measures the proportion of the expected concepts/mappings that exist in a mapping repository (i.e., M-Ontology or MO-Repository). *Precision* measures the proportion of the correctly identified concepts/mappings over the total identified ones. *Recall* measures the proportion of the correctly identified concepts/mappings over the total correct and identifiable ones. *Fmeasure* is the weighted harmonic mean of precision and recall.

Experiment 1 evaluates the performance of mapping discovery by using only M-Ontology. That means only the semantics-based selection is used without the history-based selection. The experiment is to match two query forms (from the data set AIR) that are not in M-Ontology. As illustrated by red dotted lines in Fig. 4, our ontology-based approach to mapping discovery looks promising in both feasibility and effectiveness. First, Fig. 4(a) shows that Hit-Rate reaches almost the highest value 1.0 when m is above 10. That indicates that most of

mappings can be found from M-Ontology when at least 10 query forms are fully integrated in meta-queriers. That is, the reuse of mappings is practical. Second, in Fig. 4(b), we observe Fmeasure is above 0.8 if m is larger than 5 and above 0.9 when m is larger than 35. As expected, the effectiveness of our mapping discovery algorithm improves with the M-Ontology enrichment. When there exists enough information in M-Ontology, most of mappings can be effectively identified by our algorithm. The following two experiments will present more observations and analyses.

Experiment 2 evaluates the performance of semantics-based element classification, which is a core operation in mapping discovery through M-Ontology. This operation is to classify the elements in a specific query form (from the data set AIR) to appropriate concept nodes based on their similarity with the representative objects of these concept nodes. The corresponding performance is shown by blue solid lines in Fig. 4. As illustrated, the similar trends can be observed for both element classification and mapping discovery. Discovery of correct mappings requires exact classification of the elements in both query forms. Thus, ideally, element classification should perform apparently better than mapping discovery with respect to Hit-Rate and Precision and worse with respect to Recall. However, the difference of Recall values is relatively minor, compared with the corresponding precision values, especially when m is smaller than 10 and larger than 35. Based on our analysis, the major reason is that some concept nodes appear only in few (more than one) schemas. The schema elements that are hooked to these concept nodes often do not correspond to any element in the query form that is to be matched. The total number of target mappings is lower than the average of schema elements in our data sets. Thus, the recall difference is not large. The existence of such concept nodes can be identified at the Fig. 4(a). When m is above 10, almost all the mappings are able to be found, but still more than 10% schema elements cannot be hooked to the concept nodes. In our experiments, we do not consider the mismatches caused by T-Edges incompleteness and errors. The performance of mapping discovery might be worse than the number shown in Experiment 1, but we also expect the performance can be improved through manual enrichment on M-Ontology. For example, humans can manually correct the auto-generated information in the representative objects (e.g., representative labels).

Experiment 3 examines the effects of query-form evolution on the discovery through M-Ontology without history-based element classification. The data set used in Experiments 1 and 2 is AIR, which consists of the original query forms and the evolved ones. To identify the possible influence of inclusion of the evolved forms, we conduct another set of experiments for element classification using the data set AIRO (without the evolved forms). As illustrated in Fig. 5, the blue solid lines and the green dashed lines represent the performance values of element classification respectively using AIR and AIRO. All these four charts indicate that these curve lines almost coincide. Although the similarities in terms of design, naming and functionality can be observed between the original query forms and the evolved ones, our semantics-based classification ignores these similarities and does not gain benefits from them.

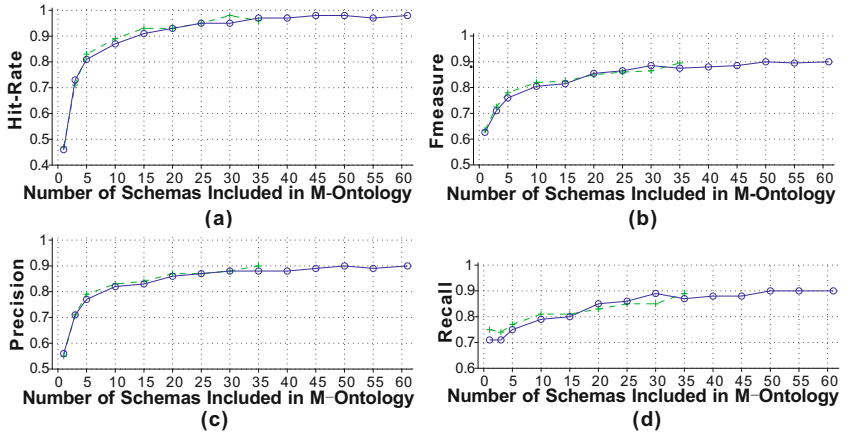


Fig. 5. Experiment results of concept searching for schema elements

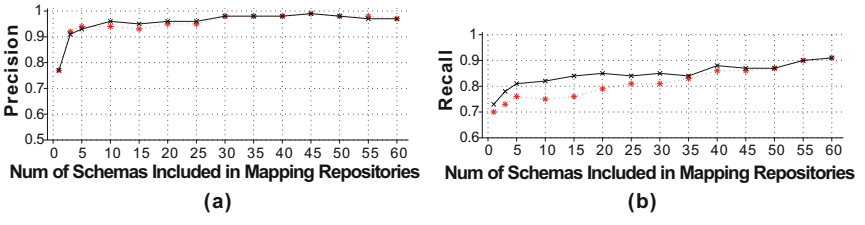


Fig. 6. Experiment results of mapping discovery through M-Ontology and MO-Repository

Experiment 4 is to evaluate the performance improvement with the usage of MO-Repository. Due to limits of the real-world data sets (AIR), there does not exist enough evolved mappings to obtain a complete evaluation. However, we still can conduct a set of experiments to show the benefits gained from the function matchEtoE, which returns a set of element correspondences from MO-Repository. These correspondences are used in the history-based element classification. Thus, the partial benefits still can be shown as illustrated in Fig. 6. The red dotted lines and black solid lines represent the performance values of mapping discovery respectively using only M-Ontology and both repositories. They share the same experiment samples. We see a significant increase in Recall and almost no change in Precision when less than 25 query forms are integrated into the mapping repositories. That means more correct mappings are found without loss of precision. If the number of integrated query forms is larger than 25, the Recall increase appears modest and even negligible. That is, most mappings can be discovered using M-Ontology (without MO-Repository), when sufficient mappings have been inserted.

From the above four sets of experiments, we observe several important properties of our reuse-oriented mapping discoverer. The data set AIR from real-world query forms shows almost all the mappings can be found in the mapping repositories, when there exist sufficient T-Edges to connect G/A-Nodes. Our proposed

reuse-based solution has a capability of effectively discovering most of mappings, as demonstrated in the promising results of Precision, Recall and Fmeasure. We expect better performance values can be obtained through manual correction on mapping repositories.

5 Related Work

Although schema matching has been widely researched for thirty years, most of the existing solutions (as surveyed in [11] [22]) only consider one-to-one mappings. In fact, many-to-many mappings are pervasive in real-world applications. Current solutions to complex mappings can be classified to three categories as follows:

Learning-based approaches learn mapping templates from the existing mapping instances. iMAP[7], also called COMAP[9], is proposed to solve several types of 1-to-n complex mappings by comparing the content or statistical properties of data instances from two schemas. However, the number of its hard-coded rules limits the types of complex mappings that can be found. HSM[25] and DCM[13] employ correlation-mining techniques to find the grouping relations (i.e., co-occurrence) among elements, but their solutions do not take into account how these elements correspond to each other (i.e., mapping expressions).

Template-based approaches search the most appropriate mapping templates/ rules from a template/rule set. The main drawback is their limited capability of finding complex mappings. The templates/rules are separate and static. In addition, the template number is normally very limited. IceQ[26] integrates two human-specified rules into their systems for partially finding Part-of and Is-A type mappings (i.e., two common one-to-many mappings). QuickMig[10], as an extension of COMA[3], summarizes ten common mapping templates, some of which are complex mappings. Several possible combinations of templates are also presented. It also designs an instance-level matcher for finding splitting or concatenation relationships.

Ontology-based approaches employ external ontologies for mapping discovery. Two schemas to be matched are first matched to a single ontology separately, and then element-level mappings can be found by deducing from the intermediary ontology. However, its performance is largely affected by the coverage and modeling of ontologies. For example, the ontology defined in SCROL[23] does not consider the syntactic heterogeneity (defined in Chapter 4.3) caused by various semantics-equivalent representations. In addition, the existing ontology-based solutions [27] [23] simply assume that the shared ontologies have already been built, but building such well-organized ontologies is as hard as finding complex mappings. Ontology construction is also a labor-intensive, time-consuming and error-prone problem [8] [21] [24].

Our proposed mapping discovery algorithm is a hybrid solution. First, the evolution process of a specific mapping can be viewed as a template/rule. In a sense, the discovery through MO-Repository can be viewed as finding rules to apply. Second, discovery through M-Ontology is a typical ontology-based approach. We provide an integrated solution to three indispensable subproblems: ontology design, ontology construction and mapping discovery. The design

of M-Ontology addresses five kinds of heterogeneity: language heterogeneity, syntactic heterogeneity, instance heterogeneity, structural heterogeneity and semantic heterogeneity. Third, ontology construction is a learning-based approach, i.e. incremental representative-based clustering. Different from the classical ontology construction [8] [21], the proposed ontology is generated from schemas and mappings, which are abundant in our proposed meta-querier customization. As more schemas and mappings are inserted into the ontology, more mappings can be correctly discovered from the ontology.

6 Conclusions and Future Work

Current research proves that mapping discovery cannot be fully automated in the foreseen future, especially for those many-to-many mappings with expressions. We believe that reuse of existing mappings is one of the best (or maybe the only) solutions. In our proposed solution, reusing similar mappings in the same domain can avoid the repetitive tasks when a large number of customized meta-queriers need to be built and maintained. To enhance reuse potential from the existing mappings, we introduce the ontology-based and change-oriented mapping models. Based on these two models, a user-friendly discovery algorithm is provided. In our solution, discovery of many-to-many mappings is converted to finding of one-to-one correspondences from their evolution processes and a domain-based mapping ontology. The proposed algorithms for finding correspondences are straightforward to non-technical users. Our experimental results on real-world data sets confirm the feasibility and effectiveness of our reuse-oriented solution.

Our plans for future work include: 1) Extending our M-Ontology with a strong emphasis to decrease the degree of informality in the collaboration platform, which is absolutely necessary in collaborative data integration. 2) Supporting version mechanisms and tackling the issue of error protection through coordinated mass collaboration and ontology verification mechanisms. 3) Integrating more one-to-one matching algorithms [11] [22] into our collaborative framework for mapping discovery.

References

1. HTML 4.01 specification: Forms (1999), http://www.w3.org/TR/html4/interact/forms.html
2. The UIUC Web integration repository (2003), http://metaquerier.cs.uiuc.edu/repository
3. Aumueller, D., Do, H.-H., Massmann, S., Rahm, E.: Schema and ontology matching with coma++. In: SIGMOD Conference, pp. 906–908 (2005)
4. Chang, K.C.-C., He, B., Li, C., Patel, M., Zhang, Z.: Structured databases on the Web: Observations and implications. SIGMOD Record 33(3), 61–70 (2004)
5. Chang, K.C.-C., He, B., Zhang, Z.: Toward large scale integration: Building a MetaQuerier over databases on the Web. In: CIDR, pp. 44–55 (2005)
6. Chapuis, O., Labrune, J.-B., Pietriga, E.: Dynaspot: speed-dependent area cursor. In: CHI (2009)
7. Dhamankar, R., Lee, Y., Doan, A., Halevy, A., Domingos, P.: iMAP: discovering complex semantic matches between database schemas. In: SIGMOD Conference, pp. 383–394 (2004)

8. Ding, Y., Foo, S.: Ontology research and development. part 1 - a review of ontology generation. Journal of Information Science 28(2), 123–136 (2002)
9. Doan, A.: Learning to map between structured representations of data. PhD thesis, University of Washington (2002)
10. Drumm, C., Schmitt, M., Do, H.H., Rahm, E.: Quickmig: automatic schema matching for data migration projects. In: CIKM, pp. 107–116 (2007)
11. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag New York, Inc., Heidelberg (2007)
12. Halevy, A.Y., Rajaraman, A., Ordille, J.J.: Data integration: The teenage years. In: VLDB, pp. 9–16 (2006)
13. He, B., Chang, K.C.-C.: Automatic complex schema matching across Web query interfaces: A correlation mining approach. ACM Trans. Database Syst. 31(1), 346–395 (2006)
14. He, H., Meng, W., Yu, C.T., Wu, Z.: Automatic integration of Web search interfaces with WISE-Integrator. VLDB J. 13(3), 256–273 (2004)
15. He, H., Meng, W., Yu, C.T., Wu, Z.: Constructing interface schemas for search interfaces of web databases. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 29–42. Springer, Heidelberg (2005)
16. Hong, J., He, Z., Bell, D.A.: Extracting Web query interfaces based on form structures and semantic similarity. In: ICDE, pp. 1259–1262 (2009)
17. Li, X., Chow, R.: An ontology-based mapping repository for meta-querier customization. In: SEKE, pp. 325–330 (2010)
18. Li, X., Chow, R.: Ontology-centric source selection for meta-querier customization. In: IKE (2011)
19. Melnik, S.: Generic Model Management: Concepts and Algorithms. PhD thesis, University of Leipzig (2004)
20. Moscovich, T., Chevalier, F., Henry, N., Pietriga, E., Fekete, J.-D.: Topology-aware navigation in large networks. In: CHI (2009)
21. Pinto, H.S.A.N.P., Martins, J.P.: Ontologies: How can they be built? Knowl. Inf. Syst. 6(4), 441–464 (2004)
22. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. 10(4), 334–350 (2001)
23. Ram, S., Park, J.: Semantic conflict resolution ontology (SCROL): An ontology for detecting and resolving data and schema-level semantic conflicts. IEEE Trans. Knowl. Data Eng. 16(2), 189–202 (2004)
24. Sabou, M., Wroe, C., Goble, C.A., Stuckenschmidt, H.: Learning domain ontologies for semantic Web service descriptions. J. Web Sem. 3(4), 340–365 (2005)
25. Su, W., Wang, J., Lochovsky, F.H.: Holistic schema matching for web query interfaces. 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. 77–94. Springer, Heidelberg (2006)
26. Wu, W., Yu, C., Doan, A., Meng, W.: An interactive clustering-based approach to integrating source query interfaces on the deep Web. In: SIGMOD Conference, pp. 95–106 (2004)
27. Xu, L., Embley, D.W.: Discovering direct and indirect matches for schema elements. In: DASFAA, pp. 39–46 (2003)
28. Zhang, Z., He, B., Chang, K.C.-C.: Understanding Web query interfaces: Best-effort parsing with hidden syntax. In: SIGMOD Conference, pp. 107–118 (2004)
29. Ziegler, P., Dittrich, K.R., Hunt, E.: A call for personal semantic data integration. In: ICDE Workshops, pp. 250–253 (2008)

Attribute Grammar for XML Integrity Constraint Validation*

Béatrice Bouchou[1], Mirian Halfeld Ferrari[2], and Maria Adriana Vidigal Lima[3]

[1] Université François Rabelais Tours, Laboratoire d'Informatique, France
[2] LIFO - Université d'Orléans, Orléans, France
[3] Faculdade de Computação, Universidade Federal de Uberlândia, MG, Brazil

Abstract. The main contribution of this paper is a generic *grammarware* for validating XML integrity constraints. Indeed, we use an attribute *grammar* to describe XML documents and constraints. We thus explain the main parts of this novel algorithm and we report on experiments showing that our method allows for an effective and efficient validation of XML functional dependencies (XFD).

1 Introduction

This paper deals with integrity constraint validation on XML documents. Our validation method can be seen as a *grammarware*, since it is based on a grammar describing an XML document to which we associate attributes and semantic rules. Our grammar is augmented by semantic rules that define, for each integrity constraint, the verification process. In this way we show that XML integrity constraints can be compiled to an attribute grammar [1,15]. To instantiate an integrity constraint we introduce a set of finite state automata (FSA). Indeed, XML integrity constraints are defined by using path expressions which can be seen as simplified regular expressions over XML labels. These finite state automata help us to determine the role of each node in a constraint satisfaction.

To explain the main parts of our method, we focus on the validation of functional dependencies (XFD). The approach presented here implements the general proposal introduced in [7], where we present a homogeneous formalism to express different kinds of integrity constraints, including XFDs, and introduce the basis of our general validation method.

Our method validates an XML tree in one tree traversal. In the document reading order, we first go top-down until reaching some leaves and then, bottom-up, as closing tags are reached. During the top-down visit, the validation process uses attributes to specify the role of each node with respect to a given integrity constraint. In the bottom-up visit the values concerned by the constraints are pulled up via some other grammar attributes. Its running time is linear in the size of the XML document, in the number of paths composing the constraint and in the number of obtained tuples containing the constraint values.

* Partially supported by: Codex ANR-08-DEFIS-04.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 94–109, 2011.

Related work: Although several approaches for XML functional dependencies have been proposed in the literature (we refer to [3,11,18,19,21] as examples of ongoing works on the subject over the decade), the implementation of constraint validators has received less attention. Proposals in [4,5,8] just address specific constraints such as primary or foreign keys. Our approach performance is comparable to implementations in [17,16], but contrary to them, it intends to be a *generic model for XML constraints validation*, provided constraints are defined by paths. The ideas guiding this work are the ones outlined in [7,10]. We describe an incremental validation method for keys in [6]. In [9] the notion of incremental validation is considered via the static verification of functional dependencies with respect to updates. However, in that work, XFDs are defined as tree queries which augments considerably the complexity of an implementation.

Indeed, one important difference among all the XFD proposals concerns the expressive power of the language used to specify the components of the dependency. Even if path languages in [2,18,19,20] are slightly different, they all express unary queries. In [2,18,20] only simple paths are allowed while in [19] simple, composed, ascendant or descendent paths are permitted. Differently, the approach in [11] uses an n-ary path language. Similarly to us, in all these approaches, once a path is defined, one needs to determine instantiations of this path. The way it is done depends on the path language used and on the assumption of an underlying schema (that allows to define tree tuples [2] and generalized tree tuples [20]). Notice also that paths defining constraints induce a notion of tree pattern that the document must conform to (called *Paths(D)* in [2], *Schema Graph* in [12] or *Legal paths* in [18]), which can be seen as a *schema* (though less restrictive than a DTD). Thus, following [15] and [13], an extended attribute grammar can always be used to represent the selecting part of any of these different notions of XML functional dependencies. Then, functions for checking features of the selected components must be defined. In this way, attribute grammars are generic enough to accommodate many kind of integrity constraints. For instance we plan to consider the possibility of using it to compute XFD defined by tree queries such as in [9].

Paper organisation: Section 2 introduces our XFD definitions. In Section 3 we explain how finite state automata can help us to develop our validation algorithm. Section 4 presents the XFD verification process based on attribute grammar. In Section 5 we consider complexity and experimental results.

2 Functional Dependencies in XML

Let $\Sigma = \Sigma_{ele} \cup \Sigma_{att} \cup \{data\}$ be an alphabet where Σ_{ele} is the set of element names and Σ_{att} is the set of attribute names. An XML document is represented by a tuple $\mathcal{T} = (t, type, value)$. The tree t is the function $t : dom(t) \rightarrow \Sigma$ where Σ is a set of tags and where $dom(t)$ is the set of positions $u.j$, such that $(\forall j \geq 0)\ (u.j \in dom(t)) \Rightarrow (\forall i\ \ 0 \leq i < j)\ (u.i \in dom(t))$; where i and $j \in \mathbb{N}$ and $u \in U$ (U is a set of sequences of symbols in $\mathbb{N}$, and the symbol ε which is the empty

sequence). Given a tree position p, function $type(t, p)$ returns a value in $\{data, element, attribute\}$. Similarly, $value(t, p) = \begin{cases} p & \text{if } type(t, p) = element \\ val \in \mathbf{V} & \text{otherwise} \end{cases}$ where $\mathbf{V}$ is an infinite recursively enumerable domain. We also recall that, in an XML tree, attributes are unordered while elements are ordered. As many other authors, we distinguish two kinds of equality in an XML tree, namely, *value and node equality*. Two nodes are *value equal* when they are roots of isomorphic subtrees. Two nodes are *node equal* when they are the same position. To combine both equality notions we use the symbol E, that can be represented by V for value equality, or N for node equality.

Figure 1 illustrates an XML document that models the projects of a company. Notice that each node has a position and a label. For instance, $t(\epsilon) = bd$ and $t(1.0) = pname$. Nodes in positions 1.2.1.1 and 0.1.2.1 are value equal, but nodes 0.1.2 and 1.2.1 are not value equal (element quantity in their subtrees is associated to different data values).

A path for an XML tree t is defined by a sequence of tags or labels. The path languages PL_s (defined by $\rho ::= \ l \mid \rho/\rho \mid \_$) and PL (defined by $\upsilon ::= [\,] \mid \rho \mid \upsilon//\rho$) are used to define integrity constraints over XML trees.

In PL and PL_s, $[\,]$ represents the empty path, l is a tag in Σ, the symbol "/" is the concatenation operation, "//" represents a finite sequence (possibly empty) of tags, and "_" is any tag in Σ. The language PL_s describes a path in t, while PL is a generalization of PL_s including "//". Then, one path in PL describes a set of PL_s paths. In this work we adopt the language PL that is a common fragment of regular expressions and XPath. A path P is **valid** if it conforms to the syntax of PL_s or PL and for all tag $l \in P$, if $l = data$ or $l \in \Sigma_{att}$, then l is the last symbol in P. We consider that a path P defines a finite-state automaton A_P having XML labels as its input alphabet.

Definition 1. Instance of a path P over t : Let P be a path in PL, A_P the finite-state automaton defined according to P, and $L(A_P)$ the language accepted by A_P. Let $I = v_1/\ldots/v_n$ be a sequence of positions such that each v_i is a direct descendant of v_{i-1} in t. Then I is an instance of P over t if and only if the sequence $t(v_1)/\ldots/t(v_n) \in L(A_P)$. □

As an example, consider the path $bd/project/supplier$. From Figure 1, we can see that $\epsilon/0/0.1$ or $\epsilon/1/1.1$ are instances of this path. Integrity constraints in XML are expressed by sets of paths. A set of paths can form a *pattern* M if all paths have a common prefix and for all path $P \in M$, if P_1 is a subpath of P, then $P_1 \in M$. Thus, a pattern is a tree pattern.

Definition 2. Pattern and Pattern Instance: A *pattern* is a finite set of *prefix-closed* paths in a tree t. Let $Long_M$ be the set of paths in M that are not prefix of other paths in M. Let $Instances(P, t)$ be the set of all instances of a path P in t. Let $PInstanceSet^i$ be the set of path instances that verifies:

1. For all paths $P \in Long_M$ there is one and only one instance $inst \in Instances(P, t)$ in the set $PInstanceSet^i$.
2. For all $inst \in PInstanceSet^i$ there is a path $P \in Long_M$.

3. For all instances $inst$ et $inst'$ in $PInstanceSet^i$, if $inst \in Instances(P,t)$ and $inst' \in Instances(P',t)$, then the longest common prefix of $inst$ and $inst'$ is an instance of path Q in t, where Q is the path with the longest common prefix for P and P'.

An instance of a pattern M is a tuple $I = (t^i, type^i, value^i)$ where $type^i(t^i,p) = type(t,p)$, $value^i(t^i,p) = value(t,p)$ and t^i is a function $\Delta \rightarrow \Sigma$ in which:

- $\Delta = \bigcup_{inst \in PInstanceSet^i} \{p \mid p \text{ is a position in } inst\}$
- $t^i(p) = t(p), \forall p \in \Delta$ □

A functional dependency in XML (XFD) is denoted $X \rightarrow Y$ (where X and Y are sets of paths) and it imposes that for each pair of tuples[1] t_1 and t_2 if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$. We assume that an XFD has a single path on the right-hand side and possibly more than one path on the left-hand side - generalizing the proposals in [3,18,14,19]. The dependency can be imposed in a specific part of the document, and, for this reason, we specify a *context path*.

Definition 3. XML Functional Dependency: Given an XML tree t, an XML functional dependency (XFD) is an expression of the form

$$\gamma = (C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\}\ \rightarrow Q\ [E]))$$

where C is a path that starts from the root of t (*context path*) ending at the *context node*; $\{P_1, \ldots, P_k\}$ is a non-empty set of paths in t and Q is a single path in t, both P_i and Q start at the context node. The set $\{P_1, \ldots, P_k\}$ is the left-hand side (LHS) or determinant of an XFD, and Q is the right-hand side (RHS) or the dependent path. The symbols $E_1, \ldots, E_k, E$ represent the equality type associated to each dependency path. When symbols E or $E_1, \ldots, E_k$ are omitted, value equality is the default choice. □

Definition 4. XFD Satisfaction: Let $\mathcal{T}$ be an XML document, $\gamma = (C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\}\ \rightarrow Q\ [E]))$ an XFD and let M be the pattern $\{C/P_1, \ldots, C/P_k, C/Q\}$. We say that $\mathcal{T}$ satisfies γ (noted by $\mathcal{T} \models \gamma$) if and only if for all I^1_M, I^2_M that are instances of M in $\mathcal{T}$ and coincide at least on their prefix C, we have: $\tau^1[C/P_1, \ldots, C/P_k] =_{E_i, i \in [1 \ldots k]} \tau^2[C/P_1, \ldots, C/P_k] \Rightarrow \tau^1[C/Q] =_E \tau^2[C/Q]$ where τ^1 (resp. τ^2) is the tuple obtained from I^1_M (resp. I^2_M), cf. preceding footnote. □

Notice that our XFD definition allows the combination of two kinds of equality (as in [19]). We consider some XFDs, verified by the document in Figure 1:

XFD_1: $(db, (\ \{/project/pname\} \rightarrow /project\ [N]\ ))$
Project names are unique and identify a project. The context is *db*, so in this case the dependency must be verified in the whole document.

XFD_2: $(db, (\ \{/project/pname\} \rightarrow /project))$
Subtrees of projects which have the same name are identical.

XFD_3 : $(db/project, (\{/supplier/@sname, /supplier/component/@cname\}$
$\rightarrow /supplier/component/quantity\ ))$

[1] Tuples formed by the values or nodes found at the end of the path instances of X and Y in a document T.

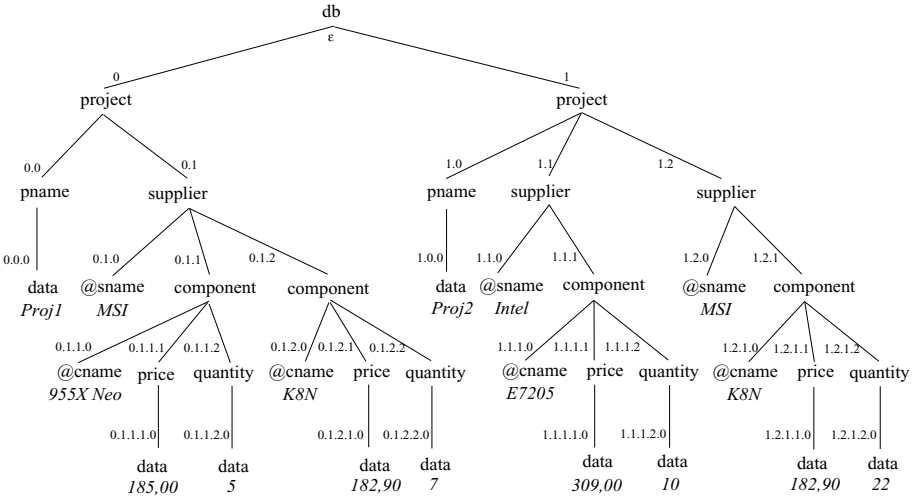


Fig. 1. Tree representing an XML document containing projects information

3 Finite State Automata for XFD

To model the paths of an XFD, we use finite-state automata (FSA) or transducers (FST). The use of finite state machines allows (i) to clearly distinguish each path (e.g. the context path) and so to define the computation of needed attributes, and (ii) to easily deal with the symbol // and thus to deal with different instantiations for a unique path (*e.g.*, instances $a.b$ and $a.x.b$ for path $a//b$).

The input alphabet of our finite machines is the set of XML tags. The output alphabet of our transducers is composed by our equality symbols. As usual, we denote a FSA by 5-tuple $A = (\Theta, V, \Delta, e, F)$ where Θ is a finite set of states; V is the alphabet; $e \in \Theta$ is the initial state; $F \subseteq \Theta$ is the set of final states; and Δ: $\Theta \times V \rightarrow \Theta$ is the transition function. A FST is a 6-tuple $A = (\Theta, V, \Gamma, \Delta, e, F, \lambda)$ such that: (i) $(\Theta, V, \Delta, e, F)$ is a FSA; (ii) Γ is an output alphabet and (iii) λ is a function from F to Γ indicating the output associated to each final state.

From Definition 3 we know that in an XFD, path expressions C, P_i and Q ($i \in [1, k]$) specify the constraint context, the determinant paths (LHS) and the dependent path (RHS), respectively. These paths define path instances on an XML tree t. To verify whether a path instance corresponds to one of these paths we use the following automata and transducers:

- The *context automaton* $M = (\Theta, \Sigma, \Delta, e, F)$ expresses path C. The alphabet Σ is composed of the XML document tags.
- The *determinant transducer* $T' = (\Theta', \Sigma, \Gamma', \Delta', e', F', \lambda')$ expresses paths P_i ($i \in [1, k]$). The set of output symbols is $\Gamma' = \{V, N\} \times \mathbb{N}^*$ such that V (value equality) and N (node equality) are the equality types to be associated to each path. Each path is numbered because there may be more than one path

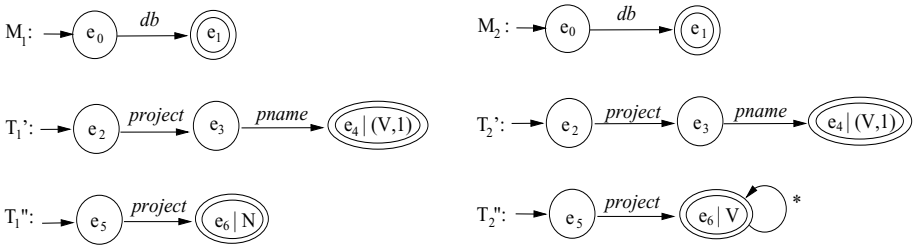


Fig. 2. Automata and transducers corresponding to XFD_1 and XFD_2

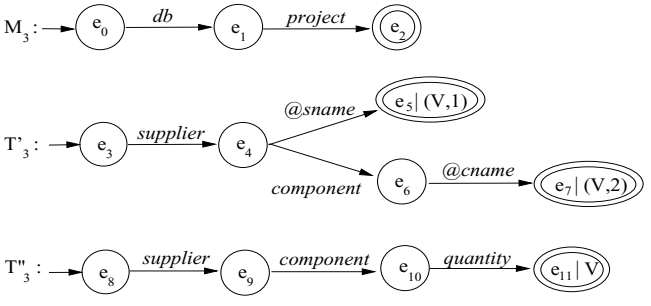


Fig. 3. Automaton and transducers for XFD_3

in the LHS. Thus, the output function λ' associates a pair $(equality, rank)$ to each final state $q \in F'$;

– Path Q is expressed by the *dependent transducer* $T'' = (\Theta'', \Sigma, \Gamma'', \Delta'', e'', F'', \lambda'')$. The set of output symbols is $\Gamma'' = \{V, N\}$ and the output function λ'' associates a symbol V or N to each final state $q \in F''$.

Figure 2 illustrates FSA and FST for constraints XFD_1 and XFD_2. We remark that the context automaton and the determinant transducers are equal for XFD_1 and XFD_2. For XFD_1, the dependent transducer expresses the application of node equality, while, for XFD_2, the application of value equality (for all values obtained from nodes rooted *project*). The automaton and the corresponding transducers for XFD_3 are illustrated in Figure 3. The determinant transducer (T'_3) gathers the attribute values of @*sname* and @*cname* that, together, determine the quantity of a component. This dependency employs value equality for @*sname*, @*cname* and *quantity* with respect to context *project*, defined by M_3.

4 XFD Validation: Attribute Grammar Approach

The integrity constraint validation process for an XML document can be accomplished with the use of an attribute grammar. Attribute grammars are extensions

of context-free grammars that allow to specify not only the syntax, but also the semantics of a language. This is done by attaching a set of semantic rules to each production of a context-free grammar. In a semantic rule, two types of attributes can be found: synthesized and inherited. Synthesized attributes carry information from the leaves of a tree to its root, while inherited ones transport information inversely, from root to leaves.

Definition 5. Attribute Grammar [1]: An attribute grammar is a triple $GA = (G, A, F)$ where: $G = (V_N, V_T, P, B)$ is a context-free grammar; A is the set of attributes and F is a set of semantic rules attached to the productions. For $X \in V_N \cup V_T$, we have $A(X) = S(X) + I(X)$, *i.e.*, $A(X)$ is the disjoint union of $S(X)$, the set of synthesized attributes of X and $I(X)$, the set of inherited attributes of X. If a is an attribute of $A(X)$, we denote it $X.a$. For a production $p : X_0 \rightarrow X_1 \dots X_n$, the set of attributes of p is denoted by $W(p) = \{X_i.a \mid a \in A(X_i), i \in [0 \dots n]\}$. For each production $p : X_0 \rightarrow X_1 \dots X_n$, the set F_p contains the semantic rules that handle the set of attributes of p. □

According to Definition 5, a set $A(X)$ of attributes is associated to each grammar symbol X to describe its semantic features. This gives rise to the following definition for the semantic rules:

Definition 6. Semantic rules attached to production rules: In an attribute grammar, each production $p : X_0 \rightarrow X_1 \dots X_n$ where $X_0 \in V_N$ and $X_i \in (V_N \cup V_T)^*$, $i \in [1 \dots n]$ is associated to a set of semantic rules of the form $b := f(c_1, c_2, ..., c_k)$, where f is a function and: (i) b is a synthesized attribute of X_0 and $c_1, c_2, ..., c_k$ are attributes of non-terminal symbols X_i, or (ii) b is an inherited attribute of a symbol X_i and $c_1, c_2, ..., c_k$ are attributes of X_0 and/or non-terminal symbols X_j, $j \in [1, \dots, i]$. □

Definition 6 establishes that the semantic analysis of a sentence using an attribute grammar is accomplished by a set of actions that is associated to each production rule. In each action definition, the values of attribute occurrences are defined in terms of other attribute values.

In the context of XFD validation, it would be possible to consider the XML type (or schema) as the grammar to be enriched with semantic rules. However, because in our approach integrity constraints are treated independently from schemas, we use a general grammar capable of describing any XML tree. Thus, we consider a context-free grammar G with the following three generic production rules where $\alpha_1 \dots \alpha_m$ denote children nodes (being either XML elements or attributes) of an element A, or the $ROOT$ element:

- Rule for the root element: $ROOT \rightarrow \alpha_1 \dots \alpha_m$, $m \in \mathbb{N}$.
- Rule for an internal element node: $A \rightarrow \alpha_1 \dots \alpha_m$, $m \in \mathbb{N}^*$.
- Rule for an element containing data and for an attribute: $A \rightarrow data$.

Grammar G is extended with semantic rules composed by attributes and actions concerning integrity constraints. Reading an XML document means visiting the XML tree top-down, opening tags, and then bottom-up, closing them. During a

top-down visit (to reach the leaves), the validation process specifies (with the aid of FSAs) the role of each node with respect to a given XFD. This role is stored in an inherited attribute. Once the leaves are reached, we start a bottom-up visit in order to pull up the values concerned by the integrity constraints. These values are stored into different synthesized attributes. In the rest of this section, Tables 1-4 introduce our attributes and semantic rules.

Inherited attribute. Firstly, we consider the inherited attribute *conf* which represents for each node in t its role concerning the given XFD. Its value is a set of FSA configurations. All nodes in t, except nodes of type *data* are bound to a *conf* attribute, but for some nodes the value of *conf* is the empty set, which means that this node is not on any path of the XFD.

Tables 1 and 2 show the semantic rules that specify the operations to be executed on *conf*, considering root or internal nodes (except leaves). The first instruction associated to production $ROOT \rightarrow \alpha_1 \ldots \alpha_m$ in Table 1 sets the attribute *conf* of the root node to be $M.q_1$, provided that the root node label is the first transition label in M. Then the value of $ROOT.conf$ is transmitted, using the descending direction, to $\alpha_i.conf$. If $ROOT.conf$ contains a configuration with a final state for M, then it is necessary to start considering the transducers for calculating configurations for $\alpha_i.conf$.

Table 1. Semantic Rule for Root Production: Attribute *conf*

Production	**Attributes**

$ROOT \rightarrow \alpha_1 \ldots \alpha_m$	$ROOT.conf := \{ M.q_1 \mid \delta_M(q_0, ROOT) = q_1 \}$
	for each α_i $(1 \leq i \leq m)$ do
	$\alpha_i.conf := \{ M.q' \mid \delta_M(q_1, \alpha_i) = q' \}$
	if $(q_1 \in F_M)$ then
	$\alpha_i.conf := \alpha_i.conf \cup \{ T'.q'_1 \mid \delta_{T'}(q'_0, \alpha_i) = q'_1 \}$
	$\cup \{ T''.q''_1 \mid \delta_{T''}(q''_0, \alpha_i) = q''_1 \}$

Similarly, Table 2 specifies how values are assigned to attributes *conf* of internal node children (using rule $A \rightarrow \alpha_1 \ldots \alpha_m$). For each α_i, we consider each configuration $\overline{M}.q$ in the parent's *conf* ($\overline{M}$ standing for either M, T' or T"): the transition $\delta_{\overline{M}}(q, \alpha_i)$ gives us a new configuration that is stored in α_i's *conf*. Furthermore, we verify if we must change from context automaton to determinant and dependent transducers, as illustrated in Example 1.

Example 1. - We consider XFD_3 (Section 2) for a company and its projects: this dependency is depicted by automaton M_3 and transducers T'_3 et T''_3 of Figure 3. The inherited attribute *conf* is calculated from the root to the leaves as shown in Figure 4. In the root node, attribute *conf* has configuration[2] $\{M_3.e_1\}$ obtained for node labelled *db*. This configuration comes from the first transition in M_3. For the node in position 0.1 (with label *supplier*), attribute *conf* contains $T'_3.e_4$ and T"$_3.e_9$, as its parent contains a configuration with M in a final state, which denotes a context node. □

[2] To simplify notations M_3,T'_3 and T''_3 do not contain indexes in Figure 4.

Table 2. Semantic Rule for Internal Node Production: Attribute *conf*

Production	Attributes

$A \rightarrow \alpha_1 \ldots \alpha_m$	for each α_i $(1 \leq i \leq m)$ do
	for each $\overline{M}.q \in A.conf$ do
	$\alpha_i.conf := \{ \overline{M}.q' \mid \delta_{\overline{M}}(q, \alpha_i) = q' \}$
	if $(\overline{M} = M) \wedge (q \in F_M)$
	then $\alpha_i.conf := \alpha_i.conf \cup \{ T'.q'_1 \mid \delta_{T'}(q'_0, \alpha_i) = q'_1 \}$
	$\cup \{ T''.q'_1 \mid \delta_{T''}(q'_0, \alpha_i) = q'_1 \}$

Synthesized attributes. We use the ascending direction to compute synthesized attributes: the values that are part of the dependency are collected, treated and carried up to the context node. At the context nodes, these values are compared in order to verify XFD satisfaction.

For each functional dependency, with possibly many paths, there are $k + 3$ synthesized attributes, where k is the number of paths in the determinant part of the dependency (Definition 3). They are denoted by c, $inters$, dc and ds_j $(1 \leq j \leq k)$. Attribute c is used to carry the dependency validity ($true$ or $false$) from the context level to the root. Attribute $inters$ gathers (bottom-up) the values from the nodes that are in determinant and dependent path intersections. Finally, ds_j and dc are attributes for storing the values needed to verify the dependency. These values can be of type $data$ (leaves of t) or node positions, according to the XFD definition of E and E_j.

Attribute $inters$ builds the tuple $<l_1, l_2>$ where l_1 is a tuple containing the values of the determinant part and l_2 contains the value of the dependent part.

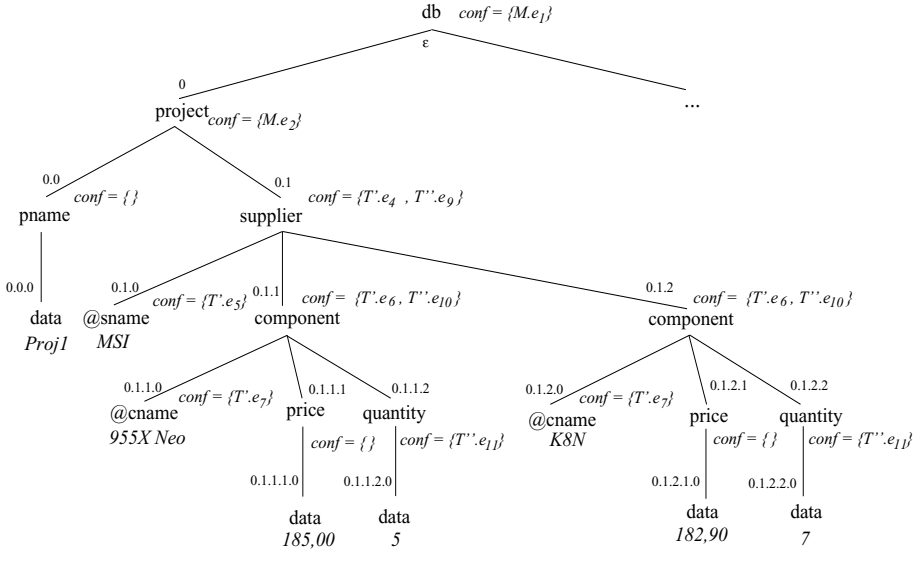


Fig. 4. Inherited attributes *conf* for XFD_3

Consider, for instance, the validation of XFD_3 in Figure 5. The context node at position 0 has just one supplier (position 0.1) that provides two distinct components. In this context, there is a path instance for *supplier/@sname* and two instances for the pattern formed by paths *supplier/component/@cname* and *supplier/component/quantity*. Thus, tuple $< l_1, l_2 >$ assigned to *inters* in position 0.1 is $\{<< MSI, 955XNeo >, 5 >, << MSI, K8N >, 7 >>\}$. To compute the value of this tuple we combine the values carried from the determinant part and assemble this combination with the value of the dependent part. Then, attribute *inters* carries the XFD values up to the context level.

Table 3. Semantic Rule for Leaf Production: Attributes ds_j and dc

Production	**Attributes**
$A \rightarrow data$	for each configuration $\overline{M}.q$ in $A.conf$ do if $(\overline{M} = T') \wedge (q \in F_{T'})$ $y := \lambda'_{T'}(q)$ $j := y.rank$ if $(y.equality = V)$ then $A.ds_j := < value(t, data) >$ else $A.ds_j := < value(t, A) >$ if $(\overline{M} = T'') \wedge (q \in F_{T''})$ if $(\lambda''_{T''}(q) = V)$ then $A.dc := < value(t, data) >$ else $A.dc := < pos(t, A) >$

The values of attributes c, *inters*, ds_j and dc are defined according to the role of the parent node *w.r.t.* the XFD. We recall that this role is given by the value of attribute *conf*. Table 3 shows how to calculate ds_j and dc for parents of nodes *data* (the grammar rule for leaves). As seen in Section 3, transducer T' has an output function λ' and associates the couple $(equality, rank)$ to each final state $q \in F'$ of T', where *equality* stores the equality type (V ou N) and *rank* is the rank j of P_j. Transducer T'' follows the same idea, but in this case, as the dependent part of an XFD has just one path, the output function λ'' associates to each final state $q \in F''$ only one symbol representing the equality type. In Table 3, the function $value(t, A)$ returns the parent *position*. In Figure 5 we depict values of attributes ds_1, ds_2 and dc for XFD_3.

Table 4 defines the synthesized attribute computation for internal nodes. In the following, we explain their computation, respecting the numbering in Table 4. We denote p the parent's position whose synthesized attributes are computed.

1. When p is a node in the XFD dependent path (transducer T'') and is also the last node, we have: if $E = V$, values for attribute dc are obtained from the values of p descendants whose type is *data*; if $E = N$ then dc stores p.
2. When p is the last node for an XFD determinant path P_j we have: if $E = V$ the value of an attribute ds_j is obtained from p descendants whose type is

Table 4. Semantic Rule for Internal Node Production: Attributes ds_j, dc, $inters$, c

Production	**Attributes**

$A \rightarrow \alpha_1 \ldots \alpha_m$	$intersFlag := true$
	for each configuration $\overline{M}.q$ in $A.conf$ do
	(1) if $(\overline{M} = T'') \wedge (q \in F_{T''})$ then
	if $(\lambda''_{T''}(q) = N)$ then $A.dc := < value(t, A) >$
	else $A.dc := < \alpha_1.dc, \ldots, \alpha_m.dc >$
	(2) if $(\overline{M} = T') \wedge (q \in F_{T'})$ then
	$y = \lambda'_{T'}(q) \qquad j = y.rank$
	if $(y.equality = N)$ then $A.ds_j := < value(t, A) >$
	else $A.ds_j :=< \alpha_1.ds_j, \ldots, \alpha_m.ds_j >$
	(3) if $(\overline{M} = T'') \wedge (q \notin F_{T''})$ then
	for each α_i $(1 \leq i \leq m)$ do
	if $(\alpha_i.inters =<>)$ then $A.dc := \alpha_i.dc$
	if $(\overline{M} = T') \wedge (q \notin F_{T'})$ then
	for each α_i $(1 \leq i \leq m)$ do
	if $(\alpha_i.inters =<>)$ then
	for each j $(1 \leq j \leq k)$ do $A.ds_j \mathrel{+}= \alpha_i.ds_j$
	(4) if $(\overline{M} = M) \wedge (q \in F_M)$
	then $A.inters := A.inters + \alpha_w.inters$
	$A.c := < \forall\, w, z$ in $A.inters, w \neq z$: $w.l_1 = z.l_1$
	$\Rightarrow w.l_2 = z.l_2 >$
	$intersFlag := false$
	(5) if $(\overline{M} = M) \wedge (q \notin F_M)$
	then $A.c := < (\forall\, x_w : \alpha_w.c =< x_w >\Rightarrow \bigwedge_{w=1}^{m} x_w) >$
	$intersFlag := false$
	end for
	(6) if $(intersFlag = true)$ then
	for each j $(1 \leq j \leq k)$
	if $(A.ds_j =<>)$ then $A.ds_j := \varepsilon$ // ε is the empty string
	if $(A.dc =<>)$ then $A.dc := \varepsilon$
	$temp := < A.ds_1 \times \cdots \times A.ds_k >$
	$A.inters := < temp \times A.dc >$
	if $(\forall\, \overline{M}.q \in A.conf : q \notin F_{T'} \wedge q \notin F_{T''})$ then
	for each α_i $(1 \leq i \leq m)$ do
	$A.inters \mathrel{+}= \alpha_i.inters$
	$A.inters := mapping(A.inters)$

data; if $E = N$ then ds_j stores p. In this case, *rank* j and the equality type of P_j come from λ' $(y = \lambda'_{T'}(q))$.

3. When p is a node corresponding to the intersection of paths in XFD, then p is seen as a point where obtained values should be combined in order to build a tuple containing the values of the determinant and the dependent parts. We denote this tuple by $< l_1, l_2 >$, where l_1 is a tuple of determinant values, and l_2 is the dependent value. As a result, $< l_1, l_2 >$ contains the combined XFDs values to be carried up.

4. When p is an XFD context node then attribute *inters* should keep all n-tuples (the final dependency values) from attributes *inters* of descendant nodes. At this point the XFD is validated for a specific context, and if it is satisfied, attribute c is assigned *true*, otherwise *false*.
5. When p is a node in the context path, then c is assigned the conjunction of values obtained from attribute c of its descendants (one attribute c for each context node). If c is *true* then the XFD is respected up to node p. If p is the root node then the XFD is respected in the whole document.
6. To combine values from the determinant and dependent parts we use a boolean variable (*intersFlag*), a variable (*temp*) and a *mapping* function. Variable *intersFlag* indicates when there are no more intersections to calculate (which means that the node is in the context path). When there are intersections to calculate, we proceed by steps. Firstly we build a n-tuple for ds_j and dc of a node A. This is done by assigning to *temp* the Cartesian product of each ds_j. Afterwards, a second Cartesian product is done to combine attribute values from *temp* and dc. The resulting tuple $< l_1, l_2 >$ is stored in $A.inters$. At this point, function *mapping* verifies, for each tuple l_1, if there are empty values that may be replaced by non-empty values obtained from another tuple l_1. It returns a completed tuple.

Example 2. In Figure 5, we show the computation of attributes c, $inters$, ds_j and dc for XFD_3 of Example 1. Due to the determinant part of XFD_3, attributes ds_j store the values obtained from $@fname$ (supplier name) and $@cname$ (component name). On the other hand, dc stores a value for *quantity*. The attributes ds_j and dc carry the dependency values up to the first intersection

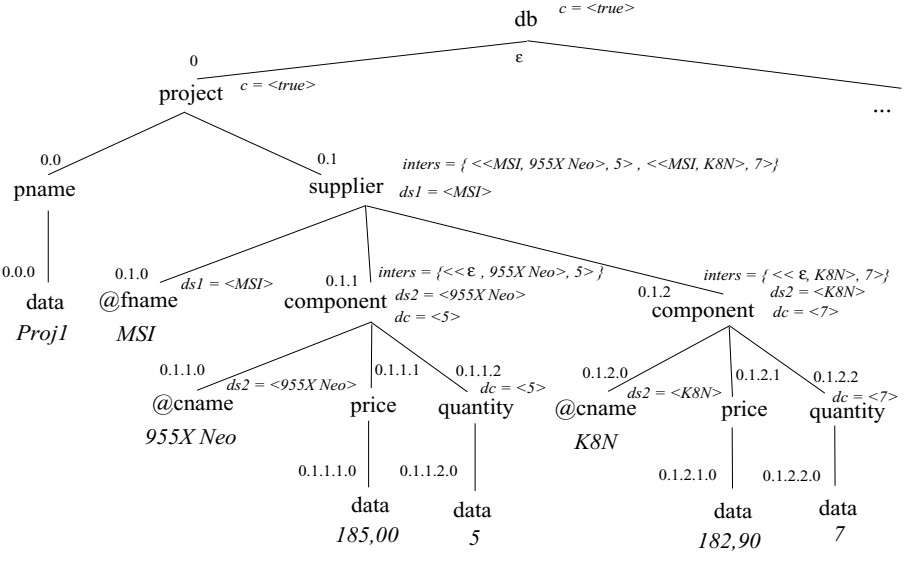


Fig. 5. Synthesized Attributes c, $inters$, ds_j and dc for XFD_3

node (*component*). Notice that for position 0.1.1 we have $ds_1 = \{\epsilon\}$ $ds_2 = \{955XNeo\}$ and $dc = \{5\}$. To compute *temp* we perform $ds_1 \times ds_2$ which gives $temp = \{\{\epsilon, 955XNeo\}\}$. The attribute *inters* is computed using Cartesian products between ds_j and dc. Thus $inters = temp \times dc$, which gives $inters = \{\{\{\epsilon, 955XNeo\}, 5\}\}$. The function *mapping* is not executed for nodes *component* (positions 0.1.1 and 0.1.2) because there is only one tuple l_1 in each attribute *inters*. The next intersection, for node *supplier* in position 0.1, creates the following tuple, obtained from ds_1 of *@sname*: $< <MSI,\varepsilon>, \varepsilon>$. The attribute *inters* of node *supplier* stores the new tuple and also puts together the tuples of attributes *inters* from subnodes: $< <<MSI,\varepsilon>, \varepsilon>,< <\varepsilon, 955XNeo>,5>,< <\varepsilon, K8N>,7> >$. Next, function *mapping* verifies for each two binary tuples in *inters* if their values can be joined. The result is: $< < <MSI, 955XNeo>,5>, < <MSI, K8N>,7> >$. In the context node, labelled *project*, the dependency is verified (according to Definition 4) and the value *true* is assigned to the attribute *c*. This last attribute is carried up to the tree root as well as attributes *c* from other context nodes. □

To finish this section, we notice that the grammar presented here generates any well-formed XML document (having elements or attributes in Σ_{elem} or Σ_{att}). It can be seen that the documents which respect a given set of XFD are exactly the ones having a value **true** as the attribute *c* for their context nodes.

5 Algorithm Analysis and Experimental Results

As we have discussed in [7,10], our *grammarware* can be regarded as a generic way of implementing constraint verification from scratch that requires only one pass on a XML document. Indeed, our way of using attribute grammar for verifying integrity constraints consists in the following stages:

(1) define a generic grammar capable of generating any labelled tree;
(2) define inherited attributes to distinguish nodes which are involved in the integrity constraints, specified by using FSA;
(3) define synthesized attributes whose values are computed by functions that check the properties stated by a given constraint.

Thus, our *generic aspect* refers to the fact that, by adapting some parameters, the same reasoning is used to validate different constraints: in particular, by determining which nodes are important in a constraint definition and, as a consequence, by establishing which FSA and attributes are needed.

The following table illustrates the parameter adaptation for XFD, keys (XKeys) and foreign keys (XFK). The case XFD was discussed in this paper. We refer to [6] for details concerning key and foreign key validation. Notice that besides context and leaf nodes, key specification also needs an extra special node denoted by *target*. Consequently, three finite state automata are used, one associated to each special node in the key (or foreign key) specification.

Constr.	Path expression	FSA	Attributes
XDF	$(C, (\{P_1\ [E_1], \ldots, P_k\ [E_k]\} \rightarrow Q\ [E]))$	M, T and T'	Inherit.: : *conf* Synth.: $c, inters, ds_j, dc$
XKeys	$(C, (Tg, \{P_1, \ldots, P_k\}))$	A_C, A_{Tg} et A_P	Inherit.: *conf* Synth.: c, tg et f
XFK	$(C, (Tg^R, \{P_1^R, \ldots, P_k^R\}) \subseteq (Tg, \{P_1, \ldots, P_k\}))$	A_C, A_{Tg}^R, A_P^R	Inherit: *conf* Synth.: c, tg et f

As shown in Section 4, XFD validation can be divided in two parts: (i) generation of tuples and (ii) checking, at a context level, the distinctness or (value) equivalence of the obtained tuples. Tables 3-4 have offered the details of these operations: the rules describe how to compose tuples, how to verify when we reach a context node and, in this case, how to perform appropriate checking. The validation of other integrity constraints is done in a similar way, changing the tests performed and the actions in concerned nodes.

The generation of tuples and their verification for a given XFD is done while parsing the XML document $\mathcal{T}$ and its time complexity is $\mathcal{O}(n.n_p.n_t)$ where n is the number of nodes in $\mathcal{T}$, n_p is the number of paths for the given XFD and n_t is the number of obtained tuples (instances of the XFD to be compared). This complexity is not affected by the shape of the XML document, but it can be affected by the number of XFD instances existing in the document. When there is a large number of XFD instances, the comparisons performed are time consuming at context level.

Each XFD is checked by running the finite state automaton that corresponds to its path and we use two stack structures to store the inherited and synthesized attributes. The synthesized attributes are collected to compose the XFD tuples until a context level. At this point, we use a hash table to store the formed tuples. The index/value pair for the hash table is defined by the tuple determinant and dependent parts, respectively. Thus, a tuple insertion in the hash table is valid if its determinant part is not already an index. Otherwise, the dependant value of the tuple that exists in the hash table is compared with the dependant value of the one to be inserted: if they are distinct then the XFD is not respected under the particular context and this part of the validation returns *false*.

The implementation of our validation method was done in Java. XML documents have been created specifically for our tests using the template-based XML generator ToXGene[3]. By using the Xerces SAX Parser documents are read and the necessary information stored into our data structures. The experiments were performed using a PC with Intel Pentium Dual CPU TE2180 at 2.00GHz, 2GB RAM under Microsoft Windows XP. For the tests illustrated in Figure 6, we used 4 XML documents containing projects information (as shown in Figure 1) with varying sizes (8MB, 41MB, 83MB and 125MB) where we considered strings of size 10 and integers of size 3 for random data generation in ToXGene templates. In Part (a) we show time validation when we have a fixed number of

[3] http://www.cs.toronto.edu/tox/toxgene/

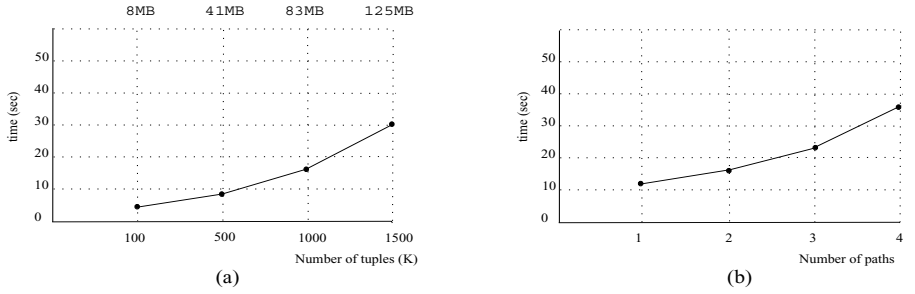


Fig. 6. (a) Validation time with number of LHS paths fixed to 2. (b) Validation time with number of tuples fixed to 1000K.

paths (2 determinant paths as XDF_3 shown in Section 2) but a varying number of concerned tuples. The validation time increases linearly *w.r.t.* the number of tuples. In Part (b) we do the inverse: we have a varying number of determinant paths for a fixed number of tuples (1000K).

6 Conclusions

An attribute grammar can be used as an integrity constraint validator. This paper shows its application as an XFD validator while in [6] the same reasoning has been used for keys. As in [17,16], the validation is performed in linear time *w.r.t.* the document size and the number of XFD paths and instances. The added value of our proposal lies in *its generic nature*, since our generic attribute grammar can stand for any XML constraint validator (provided that the constraint is expressed by paths), by adjusting attributes, tests and the needed FSA. An incremental version of XFD validation is obtained by extending the proposals introduced in [6]. We consider the possibility of adapting our approach to implement more powerful languages such as the tree patterns proposed in [9].

References

1. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: principles, techniques, and tools. Addison-Wesley, Reading (1988)
2. Arenas, M., Libkin, L.: A normal form for XML documents. In: ACM Symposium on Principles of Database System (2002)
3. Arenas, M., Libkin, L.: A normal form for XML documents. ACM Transactions on Database Systems (TODS) 29(1) (2004)
4. Benedikt, M., Bruns, G., Gibson, J., Kuss, R., Ng, A.: Automated update management for XML integrity constraints. In: Program Language Technologies for XML, PLANX 2002 (2002)
5. Bidoit, N., Colazzo, D.: Testing XML constraint satisfiability. Electr. Notes Theor. Comput. Sci. 174(6), 45–61 (2007)

6. Bouchou, B., Cheriat, A., Halfeld Ferrari, M., Laurent, D., Lima, M., Musicante, M.: Efficient constraint validation for updated XML databases. Informatica 31(3), 285–310 (2007)
7. Bouchou, B., Halfeld Ferrari, M., Lima, M.: Contraintes d'intégrité pour XML. visite guidée par une syntaxe homogéne. Technique et Science Informatiques 28(3), 331–364 (2009)
8. Chen, Y., Davidson, S., Zheng, Y.: XKvalidator: A constraint validator for XML. In: Proceedings of ACM Conf. on Information and Knowledge Management (2002)
9. Gire, F., Idabal, H.: Regular tree patterns: a uniform formalism for update queries and functional dependencies in XML. In: EDBT/ICDT Workshops (2010)
10. Halfeld Ferrari, M.: Les aspects dynamiques de XML spécification des interfaces de services web avec PEWS. Habilitation á diriger des recherches, Université François Rabelais de Tours (2007)
11. Hartmann, S., Link, S., Trinh, T.: Solving the implication problem for XML functional dependencies with properties. In: Dawar, A., de Queiroz, R. (eds.) WoLLIC 2010. LNCS, vol. 6188, pp. 161–175. Springer, Heidelberg (2010)
12. Hartmann, S., Trinh, T.: Axiomatising functional dependencies for XML with frequencies. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 159–178. Springer, Heidelberg (2006)
13. Koch, C., Scherzinger, S.: Attribute grammars for scalable query processing on XML streams. The VLDB Journal 16, 317–342 (2007)
14. Liu, J., Vincent, M.W., Liu, C.: Functional dependencies, from relational to XML. In: Ershov Memorial Conference, pp. 531–538 (2003)
15. Neven, F.: Extensions of attribute grammars for structured document queries. In: Proceedings of International Workshop on Database Programming Languages (1999)
16. Shahriar, M. S., Liu, J.: On the performances of checking XML key and functional dependency satisfactions. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009. LNCS, vol. 5871, pp. 1254–1271. Springer, Heidelberg (2009)
17. Vincent, M.W., Liu, J.: Checking functional dependency satisfaction in XML. In: XSym, pp. 4–17 (2005)
18. Vincent, M.W., Liu, J., Liu, C.: Strong functional dependencies and their application to normal forms in XML. ACM Transactions on Database Systems 29(3) (2004)
19. Wang, J., Topor, R.: Removing XML data redundancies using functional and equality-generating dependencies. In: Proceedings of the 16th Australasian Database Conference (2005)
20. Yu, C., Jagadish, H.: XML schema refinement through redundancy detection and normalization. The VLDB Journal 17, 203–223 (2008)
21. Zhao, X., Xin, J., Zhang, E.: XML functional dependency and schema normalization. In: HIS 2009: Proceedings of the 9th International Conference on Hybrid Intelligent Systems, pp. 307–312 (2009)

Extracting Temporal Equivalence Relationships among Keywords from Time-Stamped Documents*

Parvathi Chundi, Mahadevan Subramaniam, and R.M. Aruna Weerakoon

University of Nebraska at Omaha,
NE, 68182 Omaha, NE, US
{pchundi,msubramaniam,rweerakoon}@mail.unomaha.edu

Abstract. Identifying keyword associations from text and search sources is often used to facilitate many tasks such as understanding relationships among concepts, extracting relevant documents, matching advertisements to web pages, expanding user queries, etc. However, these keyword associations change as the underlying content changes with time. Two keywords that are associated with each other during one time period may not be associated in another time period or the context under which these keywords are associated may be different. In this paper, we define an **equivalence** relationship among a pair of keywords and develop methods to construct a temporal view of the equivalence relationship. Given a document set D, a keyword a is associated with a **context** consisting of frequently occurring keyword sets (f_s) of D in which a appears. Two keywords a and b are **equivalent** in D if their contexts are the same. We say that a and b are **temporally equivalent** in a time interval if a and b are equivalent in the documents published during that time interval. Given a time-stamped document set D published over a time period T, we define the **temporal equivalence partitioning** problem to construct a partitioning of the time period T into a sequence of maximal length time intervals such that in each time interval keywords a and b are either temporally equivalent or the equivalence relationship does not hold. A temporal equivalence partitioning of a document set for a given pair of keywords highlights all of the different contexts in which the given keywords are associated which can be used to generate time-varying keyword suggestions to users. We show the effectiveness of the approach by constructing the temporal equivalence partitionings of several pairs of keywords from the Multi-Domain Sentiment data set and the ICWSM 2009 Spinn3r data set.

1 Introduction

The information available on the Web is constantly evolving. Documents discussing new topics are created and/or older documents are constantly updated. Almost all

* This work was partially supported by NSF Grant IIS-0534616 and by Grant Number P20 RR16469 from the National Center for Research Resources (NCRR), a component of the National Institutes of Health (NIH).

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 110–124, 2011.

documents available on the Web today contain time stamps that denote the time of creation/publication of a document as well its update. These time stamps are extremely useful for temporal analysis of the changing information.

Keywords (key phrases) are central for representing the information on the Web as well as for accessing relevant information. Keywords and associations among keywords are often used to match user needs to relevant documents, select advertisements that match web pages displayed to a user, to improve user queries, etc. However, with constantly changing underlying document set, it is extremely challenging for users to know the exact keywords to use to express their information needs. As the information evolves, using the same keywords to express a user need may not result in access to relevant information since the evolved information may not contain the same keywords or may contain lesser number of them which may result in lower ranking of relevant documents. And, the keyword associations may also change in the evolved information – some keywords may no longer be associated with each other.

In this paper, we describe a novel approach to extract evolving temporal relationships among keywords from a time stamped document set. We focus on one particular relation among keywords which we call **equivalence**. We first define when a pair of keywords a and b are equivalent. We then capture the temporal changes in the equivalence of keywords by constructing a partitioning of the time period of the given document set into a sequence of maximum length intervals such that, in each interval, the equivalence relationship is preserved.

Our approach analyzes a set of documents D published over a time period T to extract equivalence among keyword pairs. We assume that T is represented as a list of time points of some base granularity day, month, etc. Each document in D is assigned to one of the time points in T based on its time stamp.

Each document is represented as set of keywords. Then, we use the frequent item set computation to compute the sets of keywords that are frequent in a document set [8]. The **context** of a keyword a in a document set is the set of all maximal frequent keyword sets that contain a. Intuitively, two keywords a and b are equivalent in a document set D if the contexts of a and b in D are the same when a is substituted by b or vice versa. In that case, we refer to the contexts of a and b as **synonymous contexts**.

Keywords a and b are **temporally equivalent** in a time point (or an interval) if a and b are equivalent in the document set of that time point (or interval). To identify temporal changes in the equivalence of a and b, we introduce the notion of an **equivalence preserving** and **non-equivalence** preserving interval. Informally, given an interval T_i, T_i is equivalence preserving for a and b if a and b are temporally equivalent everywhere in T_i with the same synonymous contexts. An equivalence preserving interval T_i is **maximal** if there is no other interval T_j that properly contains T_i and T_j is an equivalence preserving interval for a and b. An interval T_k is non-equivalence preserving for a and b if the context of a and b are unchanged throughout T_i and a and b are not equivalent anywhere in T_i. Maximal non-equivalence preserving interval can be defined similar to that as maximal equivalence preserving interval.

Given a time stamped document set D and a pair of keywords a and b that appear in D, we define the problem of **optimal temporal equivalence partitioning problem** and describe an efficient algorithm to construct it. The time period T of the given document set is partitioned into a sequence of maximal equivalence or non-equivalence intervals to highlight all the changes in the equivalence of keywords a and b. Suppose we consider consecutive intervals T_i and T_{i+1} in the partitioning. If both T_i and T_{i+1} are equivalence preserving intervals for a and b, then it must be that a and b are equivalent in these two intervals under two different synonymous contexts. If T_i and T_{i+1} are both non-equivalence preserving intervals, then the context of a or b or both are different as we move from T_i to T_{i+1} and a and b are not temporally equivalent in either T_i or T_{i+1}.

We show the effectiveness of our approach by constructing the optimal temporal equivalence partitionings of several hundred pairs of keywords from two different document sets – the DVD review subset of the Multi-Domain Sentiment data set [3] and the U.S election document set from the Spinn3r data set [7]. Our preliminary results are very encouraging. They show that, for all of the keyword pairs tried, the equivalence relationship changes with time. There are time intervals when the pair of keywords is temporally equivalent and others when they are not equivalent. When the equivalence preserving intervals were examined, for all the keyword pairs tried, the intervals spanned different lengths of the time period. Moreover, the synonymous contexts of a pair of keywords were different as the equivalence preserving intervals changed over time. The experimental results confirm our observation that equivalence of keywords changes over time and that keywords are equivalent under different context at different times.

The rest of the paper is organized as follows. In Section 2, we outline some preliminary definitions. In Section 3, we discuss the temporal equivalence of keywords and describe a novel and efficient algorithm for constructing an optimal temporal equivalence partitioning for a pair of keywords. In Section 4, we discuss some experimental results. Section 5 discusses the related work and Section 6 concludes the paper.

2 Preliminary Definitions

A *time point* is an instance of time with a given *base granularity*, such as a second, minute, day, month, year, etc. A time point can be represented by a single numerical value, specifying a given second, minute, day, etc. A time period T is a sequence of n consecutive time points $t_1, ..., t_n$. An interval T_i of T is a sequence of consecutive time points $t_{i1}, \ldots, t_{ip}$ $(1 \leq i1 \leq ip \leq n)$, starting at time point t_{i1} and ending at time point t_{ip}, in T. Let T_i and T_j be two intervals of T. T_i is *contained* in T_j if the starting time point of T_i is later than that of T_j and the ending point of T_i is earlier than that of T_j. We use the notation T_i, T_j, so on to refer to arbitrary intervals and use notation T_{ip} to refer to intervals covering specific time points t_i through t_p.

Given two intervals T_{ip} and T_{ql}, T_{ql} *immediately follows* T_{ip} if $q = p + 1$. In this case, we say that interval T_{il} is a *concatenation* of T_{ip} and T_{ql} and is denoted as $T_{il} = T_{ip} * T_{ql}$. A *partitioning*, denoted by Π, of a time period T consisting of time points $t_1, ..., t_n$, is a division of T into a sequence of non-overlapping intervals $T_1, T_2, ..., T_k$ where each interval T_{i+1} immediately follows T_i and $T = T_1 * T_2 * \ldots * T_k$. The **size** of Π, denoted by $|\Pi|$ is the number of intervals in it.

Let D be a time-stamped document set containing documents $d_1, \ldots, d_m$. Each document is associated with a time stamp indicating its time of creation/publication. Let T be the time period over which the documents are published where $T = t_1$, $\ldots$, t_n. Each document in D is assigned to the time point in T during which it was published. We use $Docs(t_i)$ ($Docs(T_i)$) to denote the set of documents in D assigned to time point t_i (time interval T_i). And, $Docs(T)$ is D.

We represent each document in D as a set of keywords. We will slightly abuse the notation and use d_i to denote the set of keywords in document d_i as well. Let α be a user-defined threshold value such that $0 < \alpha \leq 1$. We define a **frequent keyword set** f_s over D as a set of keywords where f_s is a subset of at least α ratio of documents in D. We say that a frequent keyword set f_s over D is maximal if there no frequent keyword set f_r ($s \neq r$) over D such that $f_s \subseteq f_r$. The **context** of a keyword a in a document set D, denoted by c_a, is the set of all frequent item sets containing a.

The context c_a in D is the empty set if a does not appear at least in α ratio of documents in D. These are the empty contexts. A singleton context $c_a = \{\{a\}\}$.

Example 1. Let D be a document set with 5 documents where $d_1 = \{a, b, c, d\}$, $d_2 = \{a, b, c\}$, $d_3 = \{a, d\}$, $d_4 = \{b, c, d\}$, and $d_5 = \{a, c, d\}$. Let $\alpha = 0.5$.
The frequent item sets over D are – $\{a\}$, $\{c\}$, $\{b\}$, $\{d\}$, $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, $\{c, d\}$. Contexts of keywords a, b, c, and d over D are $c_a = \{\{a\}, \{a, c\}, \{a, d\}\}$, $c_b = \{\{b\}\{b, c\}\}$, $c_c = \{\{c\}, \{a, c\}, \{b, c\}, \{c, d\}\}$, and $c_d = \{\{d\}, \{a, d\}, \{c, d\}\}$ respectively.

We use $c_x/(a \leftarrow b)$ to denote the context obtained by replacing all the occurrences of the keyword a appearing in a context c_x with the keyword b. If a does not appear in c_x then $c_x/(a \leftarrow b) = c_x$. Keywords a and b are **equivalent** in D if c_a and c_b are not empty or singleton contexts and $c_a/a \leftarrow b = c_b/a \leftarrow b$. Then, we refer to contexts c_a and c_b as **synonymous contexts** of a and b in D.

Thus, the equivalence of two keywords a and b captures the notion of substitutability of a by b (and vice versa) with respect to the contexts in which they appear.

Example 2. In the above example, keywords a and d are equivalent since $c_a/a \leftarrow d = \{\{d\}, \{c, d\}\} = c_d/a \leftarrow d$. No other pair of keywords are equivalent.

Keywords a and b are **temporally equivalent** in a time point t_i in T if they are equivalent in $Docs(t_i)$. Temporally equivalent keywords in a time interval T_i are similarly defined using the document set $Docs(T_i)$.

3 Temporal Equivalence Partitioning

Suppose we are given a document set published over a time period T. Let a and b are keywords that appear in D. A temporal equivalence partitioning of T can be constructed to explicate all different synonymous contexts over T for a and b, if they exist. We first introduce the definitions needed for defining the temporal equivalence partitioning problem.

Let T_i be an interval of T where T_i consists of time points $t_{i1}, \ldots, t_{ip}$. Let c_a and c_b be the contexts of a and b in each of the time points t_{ih} $(1 \leq h \leq p)$ and in T_i. We say that T_i is an **equivalence preserving interval** for a and b if a and b are temporally equivalent in T_i as well as in each of the time points $t_{i1}, \ldots, t_{ip}$. T_i is a **non-equivalence preserving interval** for a and b if a and b are not temporally equivalent in every time point in T_i as well as at T_i.

Interval T_i is a maximal equivalence preserving (non-equivalence preserving) interval for a and b if there exists no interval T_j such that T_j properly contains T_i and T_j is an equivalence preserving (non-equivalence preserving) interval for a and b.

A temporal equivalence partitioning of T for a and b, denoted by Π_{ab} is a partitioning $T_1 * T_2 * \ldots * T_k$ of T where each interval T_i is an equivalence preserving interval or a non-equivalence preserving interval. We say that Π_{ab} is **optimal** if its size is a minimum. It is easy to see that Π_{ab} is optimal if each interval is either a maximal equivalence preserving or maximal non-equivalence preserving interval.

An Algorithm for Constructing an Optimal Temporal Equivalence Partitioning

Constructing a temporal equivalence partitioning requires determining if the given keywords a and b have synonymous contexts in a time point or an interval. We first discuss the relationship between a context of a keyword in a time interval and that at the time points in that interval. Context of a keyword a in a document set D consists of frequent keyword sets of D in which a appears. First consider the following example.

Example 3. Suppose we have time period T with 4 time points. $Docs(t_1)$ consists of three documents d_{11}, d_{12}, and d_{13} where $d_{11} = \{a, b, x\}$, $d_{12} = \{a, b, y\}$, and $d_{13} = \{a, b, z\}$. $Docs(t_2)$ consists of three documents d_{21}, d_{22}, and d_{23} where $d_{21} = \{a, b, t\}$, $d_{22} = \{a, b, t, q\}$, and $d_{23} = \{p, q, r\}$. $Docs(t_3)$ consists of three documents d_{31}, d_{32}, and d_{33} where $d_{31} = \{a, b, t\}$, $d_{32} = \{a, q, t\}$, and $d_{33} = \{a, b, p\}$. $Docs(t_4)$ consists of three documents d_{41}, d_{42}, and d_{43} where $d_{41} = \{a, b, p\}$, $d_{42} = \{q, t, p\}$, and $d_{43} = \{a, b, t\}$.

Suppose α is 0.5. Then the frequent keyword sets are as follows – for $Docs(t_1)$, we have $\{a\}$, $\{b\}$, $\{a, b\}$, for $Docs(t_2)$ we have $\{a\}$, $\{b\}$, $\{t\}$, $\{q\}$, $\{a, b\}$, for $Docs(t_3)$ we have $\{a\}$, $\{b\}$, $\{t\}$, $\{b\}$, $\{a, b\}$, $\{a, t\}$, and for $Docs(t_4)$ we have $\{a\}$, $\{b\}$, $\{p\}$, $\{t\}$, $\{a, b\}$.

Since frequent keyword set $\{a, b\}$ appears in both $Docs(t_3)$ and $Docs(t_4)$, it is easy to see that it also appears in time interval $Docs(T_{34})$ where $T_{34} = t_3 * t_4$. Frequent keyword set $\{a, t\}$ appears only in time point t_3. However, it is also

frequent in $Docs(T_{34})$ whereas keyword set $\{p\}$ is frequent in $Docs(t_4)$ but is not frequent in $Docs(T_{34})$.

Context of a in time point t_3 is $\{\{a,b\},\{a,t\}\}$. In time point t_4, context of a is $\{\{a,b\}\}$. And it is $\{\{a,b\},\{a,t\}\}$ in T_{34}.

The above example illustrates that frequent keyword sets of document set at a time interval is some combination of frequent keyword sets at the time points in that interval. Consequently, the context of a keyword in a time interval may also be different than those at the time points in that interval. We say that an interval T_i is a **context preserving interval** for a keyword a if the context of a in T_i is the same as that of a in each of the time points in T_i. In the above example interval T_{12}, where $T_{12} = t_1 * t_2$, is a context preserving interval for a whereas T_{34} is not. An interval T_i is a maximal context preserving interval for a keyword a if there exists no T_j containing T_i such that T_j is a context preserving interval for a.

A **context preserving partitioning** of a time period T for a keyword a, denoted by Π_a^C is a concatenation of intervals $T_1 * \ldots * T_h$ where each T_i is a maximal context preserving interval for a.

Lemma 1. *Let T_i be an interval consisting of time points t_{i1}, ..., t_{ip}. If a keyword set s is frequent in the document set of each of the time points t_{iq} ($1 \leq q \leq p$), then s is a frequent keyword set in $Docs(T_i)$. If a keyword set s is not frequent in the document set of any of the time points t_{iq} ($1 \leq q \leq p$), then s is not frequent in $Docs(T_i)$.*

The above lemma can be used to construct maximal context preserving intervals for a keyword in the following manner. Let $FSU(a)$ denote the union of frequent sets containing a from all time points in T. Suppose that $FSU(a)$ is not empty. Consider a frequent keyword set f_s in $FSU(a)$. We construct a partitioning of T by merging all time points t_i such that f_s is a frequent keyword set in $Docs(t_i)$. Time points t_j where f_s is not a frequent keyword set in $Docs(t_j)$ are similarly merged. We denote this partitioning of T as $\Pi(f_s, a)$. Using the above lemma, we observe that each interval T_i in $\Pi(f_s, a)$ is a maximal interval such that either f_s is a frequent keyword set in $Docs(T_i)$ as well as all the document sets at the time points in T_i or f_s is not a frequent keyword set in $Docs(T_i)$ as well as the in the document sets at the time points in T_i. We construct a $\Pi(f_s, a)$ for each frequent set in $FSU(a)$. The context preserving partitioning of T for a, Π_a^C, is a product[1] of partitioning $\Pi(f_s, a)$ for each frequent keyword set in $FSU(a)$.

Constructing a context preserving partitioning for keywords is the key to constructing a temporal equivalence partitioning of the time period for a given pair of keywords. The following lemma proves the relationship between a context preserving and an equivalence preserving interval of a and b.

Lemma 2. *Let T_i be an interval where contexts of keywords a and b are c_a and c_b. T_i is an equivalence preserving or a non-equivalence preserving interval for a and b if and only if T_i is a context preserving interval for both a and b.*

[1] The product of a set of partitions is the coarsest partition that is a refinement of each of the partitions in the set[1].

The product of Π_a^C and Π_b^C is a partitioning of T where each interval in the partitioning is a context preserving interval for both a and b. This product identifies the time intervals where the contexts of a and b are the same as those at the time points in that interval. Hence if a and b are temporally equivalent (or non-equivalent) in an interval in the product, then they are temporally equivalent (not temporally equivalent) in each of the time points in that interval and vice versa. Hence, the product is a temporal equivalence partitioning of T for a and b. Now, it is easy to see that such a temporal equivalence partitioning is also optimal since the product of Π_a^C and Π_b^C is the coarsest possible partitioning of the refinement of Π_a^C and Π_b^C.

The following procedure outlines the steps in the construction of Π_{ab}. The procedure takes as input a time stamped document set D, two keywords a and b, an α value for computation of frequent keyword sets. The procedure first computes the frequent keyword sets for the document sets at each of the time points in T. It then extracts the contexts of a and b in the document set at each time point. We then compute the context preserving partitioning of a and that of b. We then compute the product of these two partitionings to obtain the temporal equivalence partitioning of T for a and b . We then check the equivalence of a and b in each interval T_i of the product. If a and b are temporally equivalent in T_i, then T_i is deemed as an equivalence preserving interval for a and b. Otherwise, T_i is a non-equivalence preserving interval.

Input: Time stamped document set D published over T where $T = t_1, \ldots, t_n$, keywords a and b, and α
Output: A list of time intervals in the temporal partitioning of a and b.
Begin
  Construct the frequent keyword sets from each $Docs(t_i)$, $1 \leq i \leq n$.
  Let $FSU(a)$ ($FSU(b)$) be the set of frequent keyword sets containing a (b) collected from all time points.
  Construct the context preserving partitioning of T for keyword a by first constructing a $\Pi(f_s, a)$ for each $f_s \in FSU(a)$ and constructing a product of these $\Pi(f_s, a)$.
  Construct the context preserving partitioning of T for b in a similar manner.
  Construct a product of the partitionings constructed in the previous steps and output the list of intervals as Π_{ab}.
End.

4 Experiments

We used two different data sets to study the effectiveness of the methods described in the paper. The first data set, referred to as the **Election** data set in the paper, is a small subset of the Spinn3r Data Set, provided by *Spinn3r.com*. The data set contains around 44 million blog posts made between August 1st and October 1st, 2008. This data set contains blog posts related to a number of big events including the Olympics, US presidential conventions, etc. We used a small set of keywords – *democratic, republican, palin, mccain, obama, clinton* to

select a set of blog posts related to 2008 US presidential election from this data set. Our Election data set contains around 9700 blog posts, published during the months of August and September 2008. The time period associated with the document set contains 61 time points, one for each day. Since many blog posts contained spelling errors, we used WordNet to select clean keywords to represent each document in the set. Unfortunately, this resulted in removal of some popular keywords such as *obama*.

The second data set, referred to as the **DVD Data**, is a small subset of the multi-domain sentiment data set (version 2.0) [3]. It contains product reviews taken from Amazon.com from many product types. Some product types such as books and dvds have hundreds of thousands of reviews. Others have only a few hundred. We considered the product reviews submitted for DVDs. There were about 124,000 total reviews posted during January 1999 and April 2007. Each review is represented as a set of keywords that are not stop words. There are 100 time points in the time period of this data set, one for each month.

Obtaining the context of a keyword requires the computation of the frequent keyword sets for the document set of each time point. To do so, we used the **Apriori** [9] Package. The Apriori implementation represents each document as a binary vector over keywords. To manage the computational costs as well as to remove keywords that have a low chance of being frequent, we computed the keyword set for each document set as follows. We first considered all keywords with the document frequency of at least 0.25 in each time point. We then merged all the these keywords and sorted them in the number of time points they occur in. If a keyword has at least 0.25 document frequency in many time points, it is ranked higher. Then, we considered the top n keywords to construct the document vectors for each time point. By considering frequently occurring keywords in each time point as well as across time points, we increase the chance of finding keywords with the same context in multiple time points. However, considering only top n keywords may exclude some documents that do not contain any of the chosen keywords. Hence, the frequent keyword sets will be constructed over possibly a small percentage of the document set in each time point. For example, when we chose the top 50 most frequent keywords for the DVD data set, we processed only about 6,000 documents over the entire time period of the data set. However, for the Election Data set, we included as many as 9538 documents when we chose the top 50 most frequent keywords. To control the cardinality of the context of a keyword, we limited the maximum size of any frequent keyword set to 5.

We conducted the experiments by considering the top 30 and top 50 keywords for each data set. Let us denote the set of keywords chosen for the experiment as E_k. We computed the frequent keyword sets for each time point by setting the support value to 0.15. From the frequent keyword sets, we collected the context of each keyword in E_k in the document set of each of the time points. We then constructed the context preserving partitioning for each keyword in E_k. Then, we constructed an optimal temporal equivalence partitioning of the time period for each pair of keywords in E_k. In the optimal temporal equivalence

DVD Data Set: good, great, see, best, time, really, story, love, watch, movies, people, think, life, seen, little, know, films, better, video, years, characters, back, action, worth, want, old, plot, actors, music, world

Election Data Set: people, John, think, election, like, elections, vote, know, make, campaign, president, republican, question, open, time, going, party, candidate, american, want, state, say, way, good, bush, does, voters, right, years, national

Fig. 1. Top 30 keywords for the DVD and Election Data Sets

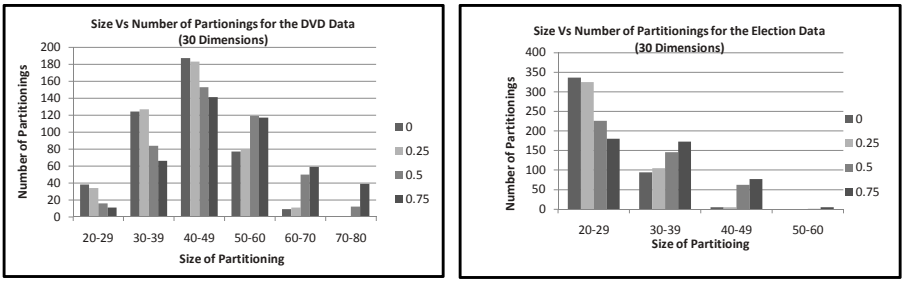


Fig. 2. Size of Π_{ab} Vs the Number of Π_{ab} with That Size for Top 30 Keywords

partitionings, we merged the consecutive non-equivalence preserving intervals into a single non-equivalence preserving interval since many of them contained empty contexts for the keywords.

In case where E_k contained the top 30 keywords, we constructed 435 temporal equivalence partitionings. When $|E_k| = 50$, it was 1176. Note that the context of a keyword may be empty in every time point for some keywords and therefore are removed from $|E_k|$. Figure 1 shows the top 30 keywords for each data set.

Given a pair of keywords a and b, constructing an optimal equivalence preserving partitioning requires checking that a and b are temporally equivalent in each of the intervals in the partitioning, which in turn needs checking that $c_a/a \leftarrow b = c_b/a \leftarrow b$. This condition was often stringent for the chosen data sets, i.e.; many keywords were often deemed non-equivalent even if the difference was simply a single singleton frequent keyword set. Therefore, the optimal temporal equivalence partitionings contained large time intervals where the keywords were not equivalent, and few (or zero) time intervals where the keywords were equivalent. Therefore, we relaxed the requirement of equivalence as follows. We say that $c_a/a \leftarrow b$ and $c_b/a \leftarrow b$ are **beta-equivalent** if the fraction of keywords that appear in some frequent keyword set in $c_a/a \leftarrow b$, but not in any keyword set in $c_b/a \leftarrow b$, and vice versa, is at most β. Using $\beta - equivalence$ allows us to treat keyword contexts as simple sets of keywords and hence checking that two contexts are beta-equivalent can also be done efficiently. Note that, if β is set to 0, then the contexts of a and b contain the same set of keywords. Higher the value of β, higher is the difference between c_a and c_b. We conducted our experiments with four different values of β, 0, 0.25, 0.5, and 0.75.

In the rest of the section, we discuss the characteristics of the optimal temporal equivalence partitionings constructed by our method. An optimal temporal equivalence partitioning of a time period T for a given pair of keywords a and b, Π_{ab}, is a partitioning of T into time intervals where each interval in the

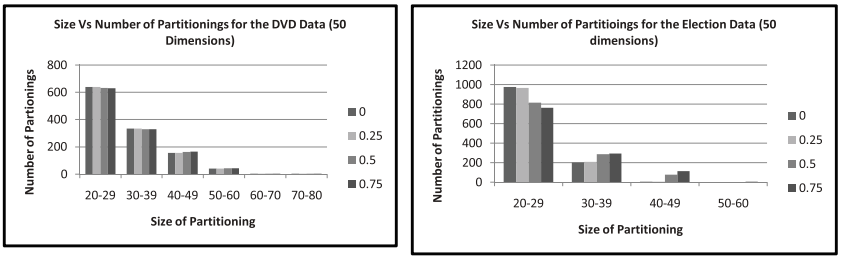


Fig. 3. Size of Π_{ab} Vs the Number of Π_{ab} with That Size for Top 50 Keywords

partitioning is equivalence or non-equivalence preserving for a and b. We analyzed the size of each Π_{ab} to understand the temporal changes in the equivalence relationship of a and b. If $|\Pi_{ab}|$ is small, there is a small number of intervals in Π_{ab}, which means that the equivalence (or non-equivalence) of a and b over time does not change much and hence can be compactly represented. In Figures 2 and 3, we show the sizes of Π_{ab} constructed, for each of the β values tried. In both figures, the X-axis plots the different ranges for sizes of Π_{ab} and the Y-axis plots the number of partitionings. For each size range on the X-axis, a set of four bars are displayed, depicting the number of partitionings that fall in that range for each value of β tried. For example, there are fewer than 50 optimal partitionings with a size in the range $20-29$ when $\beta = 0.25$ for the DVD data set when E_k consists of the top 30 keywords.

Upon examining these charts, we observed that the number of Π_{ab} in the range $20-29$ is very low when $|E_k| = 30$ for the DVD data. For that data set, most Π_{ab} were of size in the range $40-49$ and above for all values of β tried. Upon examining the data, we conjecture that, for many keyword pairs a and b, the temporal equivalence (or non-equivalence) was limited to small intervals, with length 1 or 2 time points, and could not be represented more compactly. When we expanded E_k to include top 50 keywords, the number of Π_{ab} with small sizes were very high. The number of time points for this data set is 100. Hence, Π_{ab} with size less than 50 represents the equivalence relation more compactly. Further examination of these Π_{ab}, showed that, for many keyword pairs, there were non-equivalence preserving intervals that spanned long time periods.

In case of the Election Data set, the number of Π_{ab} in each range stayed pretty much the same as we varies the cardinality of E_k. The number of time points for this data set is 61. Therefore, with many Π_{ab} in range $20-40$, we can conclude that the temporal equivalence (or non-equivalence) of a pair of keywords may span multiple consecutive time points.

To understand the effect of β, we examined the height of the bars for range. The heights of these bars change with β when E_k is 30 whereas the heights do not change much at all when E_k is 50 for both the data sets. From Figure 2, it can be observed that the size of Π_{ab} changes with β value. As β value increases, the size of Π_{ab} seems to get longer for both data sets. This is because, for small values of β, more consecutive time points were non-equivalence preserving and hence,

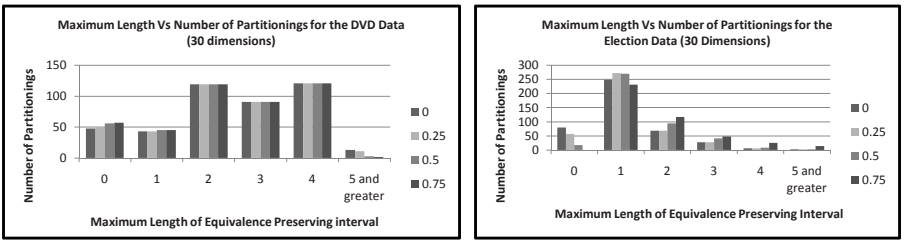


Fig. 4. Max Length of Equivalence Preserving Interval Vs the Number of Π_{ab} for Top 30 Keywords

might be represented by one long time interval whereas, as β value increased, some of the non-equivalence preserving intervals were turned into equivalence preserving intervals with non-empty synonymous contexts. Therefore, these intervals could not be combined with consecutive time points/intervals where the keyword pair was not equivalent. The effect of β was negligent when E_k is 50 for both data sets, as seen from Figure 3. This shows that β may not play any role as we increase the number of keywords used to represent each document. We will perform additional experiments to understand the effect of β on the size of Π_{ab} for large document sets.

The experiments with the size of Π_{ab} show that the equivalence of a pair of keywords changes with time, and for some keywords, it may not change too often and hence, can be compactly represented.

Next we studied the maximum length (the number of time points) of the equivalence preserving interval in the Π_{ab} of each pair of keywords a and b. For each Π_{ab} constructed, we extracted the maximum length equivalence preserving interval in the optimal partitioning. Figures 4 and 5 display these results. In the figures, the X-axis plots the different length values for the equivalence preserving intervals and Y-axis plots the number of Π_{ab}. There are 4 bars for each length value on the X-axis, one for each β value.

From Figure 4, when $|E_k|$ is 30, the maximum length of an equivalence preserving interval on average is larger for the DVD data set when compared to that of the Election data set. For many Π_{ab}, it is at least 3 time points long. For the Election data set, the maximum length of an equivalence preserving interval either 1 or 2 for most Π_{ab}. When $|E_k|$ is expanded to top 50 keywords, the maximum length of an equivalence preserving interval falls to 1 or at most 2 for many Π_{ab} in both the data sets, which can be observed from Figure 5. Although we focus only on the maximum length of an equivalence preserving interval, in general the length and where it occurs in the time period is interesting to note for any equivalence or non-equivalence preserving interval. This is because, the length gives us information about how long the context holds a keyword and the changes in it. The dates during which the equivalence preserving interval occurs on a time line is also a crucial price of information because the synonymous context associated with the interval gives clues about the reason for equivalence.

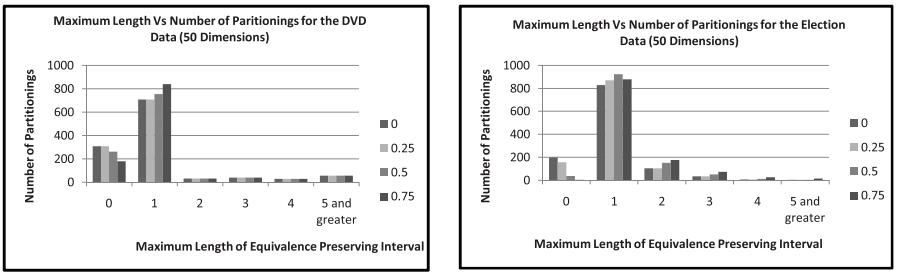


Fig. 5. Max Length of Equivalence Preserving Interval Vs the Number of Π_{ab} for Top 50 Keywords

Keyword Pairs	No. of Equivalence Preserving Intervals	Synonymous Context
movies, films	6	cinema (Aug 2006 – Sept, 2006)
Love, story	13	cinema (June – Aug 2000) mysterious, cinema, actors, watch, death (Jan 1999)
great, time	16	cinema (Feb 2001 – Aug 2001) really, best, great, time (Mar2000)
people, john	36	country, election (Aug 29th – Sep 1st) Country, election, U.S, president, republican (Aug 8th)
American, elections	2	country, u.s, john, hillary, media, bush, white, black, people (Aug 8th) Bush, country, campaign, john, time (Aug 25)
Republican, sarah	30	country, election (sep 1 – sep 6th) country, election, john (Aug 29)

Fig. 6. Synonymous Contexts Computed for Keyword Pairs

Increasing the value of β seems to have some impact on the maximum length of equivalence preserving interval for the election data. As β increases, the number of Π_{ab} with longer equivalence preserving intervals increases. On the other hand, increasing the β value has little impact on the equivalence preserving intervals for the DVD data.

In Figure 6, we display a few pairs of keywords and their synonymous contexts from one or two of the equivalence preserving intervals. The three keyword pairs were chosen for each data set. Keyword set E_k is set to top 50 keywords and β value is set to 0.5. The first column in the table shows the keyword pairs, the second column shows the total number of equivalence preserving intervals in the corresponding optimal equivalence preserving partitioning, Π_{ab}, of the keyword pair, and the third column display a couple of synonymous contexts for the keyword pair. The first row in the table shows the synonymous context for the keyword pair *movies, films*. There are 6 equivalence preserving intervals in the Π_{ab} of this pair, with the longest interval spanning *Aug 2006 - Sep 2006*. There is only one keyword in the synonymous context of *movies, films*, and that was (predictably) *cinema*. For the rest of the keyword pairs, the synonymous contexts in different equivalence preserving intervals contained overlapping keyword sets. For example, keyword pair *republican, sarah* had the longest equivalence

preserving interval from *Sep* 1^{st} to *Sep* 6^{th} with the synonymous context *country, election* whereas these two keywords are also temporally equivalent on *Aug* 29^{th} with the synonymous context *country, election, john*.

This small sample of the equivalence preserving intervals of keywords show that the proposed method finds meaningful relationships from keywords. We are currently extending the experiments to include thousands of keywords from each document set. To address scalability issues, we plan to extend the definition of equivalence of keywords to consider only the maximal frequent item sets. The notion of β-equivalence will be very useful in identifying equivalence preserving partitionings as the number of keywords are increased.

5 Related Work

Identifying keyword relationships from text documents is a classic problem in information retrieval. Keyword relationships are typically used for query expansion and document clustering [15,16]. Content based methods such as association clusters, collocation of words have been popular in identifying relationships among keywords.

Identifying keywords that represent entity names and establishing relationships among these keywords is an active area of research. We list a few references here. In [12], authors describe an approach to identify firm names and relationships between them using a search engine. Reference [5] outlines a method to extract synonymous gene and protein names from MEDLINE abstracts based on co-occurrence network. In [4], authors outline a method for identifying entities and synonyms from Wikepedia. All these approaches are non-temporal.

To the best of our knowledge, there is not much research on identifying temporally changing keyword relationships from document sets and it is an emerging research topic. In earlier works in this topic, in [6], authors describe an approach for extracting synonyms of named entities from the history of Wikipedia and classifying them into time-independent and time-dependent synonyms. The synonym relationships are then used to improve retrieval effectiveness. In [10,11], authors define several temporal relationships among keywords, such as co-occurring, ordered, and define an approach to determine these relationships. The temporal relationships are then used to generate expanded queries.

Our approach is fundamentally different from these approaches because it uses the frequent item set approach to define an equivalence relationship among a pair of keywords, which not only allows us to identify associated words at time intervals but also provide justification of associations in terms of contexts. Further, the proposed approach constructs an optimal partitioning of time period into equivalence and non-equivalence preserving time intervals highlighting all of the temporal changes in associations of the given pair of keywords.

Using frequent item sets to establish keyword relationships has also been used in [13] where word contexts based on frequent item sets were used to extract a homonym relationship among keywords. Our notion of keyword context is similar to the word contexts defined in this work. However, our application of contexts

is significantly different than that of [13]. Frequent item sets have also been used to achieve high-dimensional hierarchical document clustering (see [14] and the references contained therein.).

The notion of an optimal temporal equivalence partitioning is inspired by our earlier work on information preserving decompositions of document sets [2]. However, the proposed approach is significantly different and perhaps the first attempt at studying temporal segmentation to capture the changes in the equivalence relationship of keywords.

6 Conclusions

As the information evolves on the internet, relevant keywords and their associations change over time. In this paper, we define the problem of constructing an optimal temporal partitioning of a time period for a given pair of keywords a and b which highlights the changes in the temporal equivalence relationship of a and b. Given a document set D published over a time period T, we represent each document in D as a set of keywords. We then extract the frequent keyword sets from the document set to construct a *context* of a keyword in a document set. Two keywords are said to be *equivalent* in a document set if their contexts are the same. We then extend the notion of equivalence to time intervals. We define the notion of equivalence preserving interval of a pair of keywords a and b as follows. An interval T_i is an equivalence preserving interval for a and b if a and b are equivalent in all of T_i under the same context. The notion of non-equivalence preserving interval can be similarly defined. An optimal temporal equivalence partitioning of T for a and b is a sequence of intervals where each interval is either an equivalence preserving or non-equivalence preserving for a and b and contains the minimum possible number of intervals. We describe an efficient algorithm for constructing an optimal temporal equivalence preserving partitioning of T for a keyword pair. We implemented the proposed approach and studied its effectiveness by constructing the optimal temporal partitionings for several hundred keyword pairs obtained by analyzing the Multi-Domain Sentiment data set and the Spinn3r data set. The experiments show that the equivalence of keyword pair changes over time. And, keyword pairs are equivalent under different contexts during different time intervals. Our future work is to develop a framework to perform query expansion based on temporally changing equivalence relationship between keywords and measure the effectiveness of expanded queries. We also plan to extend the notion of temporal equivalence of a keyword pair to temporal equivalence of a set of keywords.

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge
2. Chundi, P., Rosenkrantz, D.J.: Information Preserving Time Decompositions of Time Stamped Documents. Data Mining and Knowledge Discovery Journal (July 2006)

3. Multi-Domain Sentiment Data Set (version 2.0), http://www.cs.jhu.edu/~mdredze/datasets/sentiment/
4. Bohn, C., Norvag, K.: Extracting named entities and synonyms from Wikipedia. In: IEEE Conference on Advanced Information Networking and Applications (April 2010)
5. Cohen, A.M., Hersh, W.R., Dubay, C., Spackman, K.: Using co-occurrence network structure to extract synonymous gene and protein names from MEDLINE abstracts. BMC Bioinformatics 6, 103 (2005)
6. Kanhabua, N., Norvag, K.: Exploiting Time-Based Synonyms in Searching Document Archives. In: ACM 2010 Conference on Digital Libraries (June 2010)
7. Burton, K., Java, A., Soboroff, I.: The ICWSM 2009 Spinn3r Data Set. In: International AAAI Conference on Weblogs and Social Media (May 2009)
8. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: International Conference on Very Large Data Bases (September 1994)
9. Apriori Implementation, http://www2.cs.uregina.ca/hamilton/courses/831/notes/itemsets/itemset_prog1.html
10. Kage, T., Sumiya, K.: A Web Search Method Based on the Temporal Relation of Query Keywords. In: 2006 International Conference on Web Information Systems Engineering (October 2006)
11. Kage, T., Sumiya, K.: A Temporal Clustering Method forWeb Archives. In: 22nd International Conference on Data Engineering Workshops (April 2006)
12. Jin, Y., Ishizuka, M., Matsuo, Y.: Extracting inter-firm networks from the World Wide Web using a general-purpose search engine. Online Information Review 32(2) (2008)
13. Rybinski, H., Kryszkiewicz, M., Protaziuk, G., Kontkiewicz, A., Marcinkowska, K., Delteil, A.: Discovering Word Meanings Based on Frequent item sets. In: ECML/PKDD Workshop on Mining Complex Data (September 2007)
14. Kiran, G.V.R., Shankar, K.R., Pudi, V.: Frequent Itemset based Hierarchical Document Clustering using Wikipedia as External Knowledge. In: Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (September 2010)
15. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley Longman Publishing Company, Amsterdam
16. Manning, C.D., Schutze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge

Codd Table Representations under Weak Possible World Semantics

Flavio Ferrarotti[1], Sven Hartmann[2], Van Bao Tran Le[1], and Sebastian Link[1]

[1] School of Information Management, Victoria University of Wellington, New Zealand
[2] Institut für Informatik, Technische Universität Clausthal, Germany

Abstract. Codd tables are databases that can carry Codd's null "value unknown at present" in columns that are specified as NULL. Under Levene and Loizou's possible world semantics we investigate the combined class of uniqueness constraints and functional dependencies over Codd tables. We characterize the implication problem of this class axiomatically, logically and algorithmically. Since the interaction of members in this class is intricate data engineers can benefit from concise sample tables. Therefore, we investigate structural and computational properties of Armstrong tables. These are Codd tables that satisfy the consequences of a given set of elements in our class and violate all those elements that are not consequences. We characterize when a given Codd table is an Armstrong table for any given set of our class. From this result we establish an algorithm that computes an Armstrong table in time that is at most quadratic in the number of rows in a minimum-sized Armstrong table. Data engineers can use our Armstrong tables to judge, justify, convey and test their understanding of the application domain.

1 Introduction

A database system manages a collection of persistent information in a shared, reliable, effective and efficient way. In *the relational model of data* [6] data engineers use relations to capture the structure of the application domain, and integrity constraints to restrict the relations to those considered meaningful for the application domain. Over single relation schemata the class of functional dependencies (FDs) is the most important class of integrity constraints. In particular, it subsumes the class of keys. Database management systems deviate from the relational model of data to facilitate data processing. Tables are used instead of relations. Tables may contain null values in columns declared `NULL` to model partial information. Tables may also contain duplicate rows since duplicate elimination is considered to be expensive. These features imply that the interaction of FDs over tables is more intricate than over relations.

Example 1. Consider a contact management system with column headers *Address*, *City* and *ZIP*. The combination of *Address* and *City* entries must be unique, and data entries in *City* are uniquely determined by data entries in *ZIP*. In the relational model of data we can model this semantics as the FD set Σ_1 consisting of *Address, City* $\rightarrow$ *ZIP* and *ZIP* $\rightarrow$ *City*. This can be represented concisely by a so-called Armstrong relation [13], e.g. the relation on the left.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 125–139, 2011.

Address	*City*	*ZIP*
Le Louvre	Paris	75001
Pont Neuf	Paris	75001
Pont Neuf	Toulouse	31000
Tour Eiffel	Paris	75007

Address	*City*	*ZIP*
Pont Neuf	Paris	75001
Le Louvre	Paris	75001
Pont Neuf	Toulouse	31000
`unk`	Paris	75007
Pont Neuf	`unk`	75001

Over SQL tables we capture the semantics by the set Σ_2 with the uniqueness constraint (UC) *unique*(*Address*, *City*) and the FD *ZIP* $\rightarrow$ *City*. Suppose, we specify only the column *ZIP* as `NOT NULL`. Then the table above on the right is an Armstrong table for Σ_2 and the `NOT NULL` constraints. We can see that the constraints interact very differently over tables than they do over relations. For instance, Σ_1 implies that *Address* and *ZIP* form a composite key over relations. However, over tables Σ_2 does not imply the UC *unique(Address, ZIP)*. □

For the efficient design and maintenance of real database systems it is therefore crucial to understand the interaction of UCs and FDs over tables, as illustrated by Example 1. We will model partial information by Codd's null "value unknown at present", denoted by `unk`, as adopted by SQL [8]. Hence, we will speak of Codd tables. Our class of FDs are those proposed by Levene and Loizou, and are based on a possible world semantics to allow a high degree of uncertainty.

Contributions and Organization. We summarize previous work in Section 2. The basic definitions are given in Section 3. We will characterize the interaction of UCs and FDs axiomatically, algorithmically and logically in Section 4. More precisely, we will establish a finite axiomatization that generalizes the well-known Armstrong axioms; we will establish an algorithm that decides the implication problem in time linear in the input, and we will show that it is equivalent to that of goal and definite clauses in Cadoli and Schaerf's well-known approximation logic $\mathcal{S}$-3. We then investigate structural and computational properties of Armstrong tables in Section 5. These are Codd tables that satisfy those UCs and FDs implied by a given set of such constraints and violate all others. Therefore, Armstrong tables are concise Codd table representations of UCs and FDs which data engineers can use to judge, justify, convey and test their understanding of the application domain. For Example 1 above an inspection of the Armstrong table on the right may convince the data engineers to specify the additional UC *unique*(*Address*,*ZIP*). We characterize for any given table definition T and any given set Σ of UCs and FDs in the presence of any `NOT NULL` constraints over T, when a Codd table over T is an Armstrong table. This characterization is then used to establish an algorithm for computing an Armstrong table. While the problem of finding such a table is precisely exponential in the number of column headers, our algorithm computes an Armstrong table whose number of rows is at most quadratic in the minimum number of rows required by any Armstrong table. Our results bridge the gap between the existing relational theory and database practice. We conclude in Section 6 where we also comment briefly on future work.

2 Related Work

Data dependencies and Armstrong databases have been studied thoroughly in the relational model of data, cf. [1,13]. Dependencies are essential to the design of the target database, the maintenance of the database during its lifetime, and all major data processing tasks [1,21]. Armstrong databases are a useful design aid for data engineers that can help with the consolidation of data dependencies [17] and schema mappings [2], the design of databases [19] and the creation of concise test data [9].

In relational databases, a key K over R is satisfied by a relation, if no two distinct tuples have the same values on all the attributes of K. Thus, a relation satisfies the key K if and only if the relation satisfies the FD $K \rightarrow R$. Hence, it suffices to study the class of FDs. Armstrong [3] established the first axiomatization for FDs. In general, axiomatizations can be applied by designers and administrators to validate the specification of explicit knowledge, to design and fine-tune databases or to optimize queries. An axiomatization ensures that all opportunities of utilizing implicit knowledge have been exploited. An analysis of the completeness argument can provide invaluable hints for finding algorithms that efficiently decide the implication problem. The implication problem of FDs can be decided in time linear in the input [11]. Fagin established the correspondence between the implication of functional dependencies and the implication of Horn clauses in classical propositional logic [12]. For relations, the structural and computational properties of Armstrong relations for the class of functional dependencies are well-studied [5,19].

One of the most important extensions of Codd's basic relational model [6] is incomplete information [7,16]. This is mainly due to the high demand for the correct handling of such information in real-world applications. Approaches to deal with incomplete information comprise incomplete relations, or-relations or fuzzy relations. In this paper we focus on incomplete relations. In the literature many kinds of null values have been proposed; for example, "missing" or "value unknown at present", "non-existence", "inapplicable", "no information" and "open". Several works on functional dependencies in incomplete relations exist, but none covers the case of Codd tables with arbitrary `NOT NULL` constraints. Levene and Loizou studied the class of functional dependencies over Codd *relations* where every attribute is assumed to be `NULL` [18]. Atzeni and Morfuni established an axiomatization of FDs in the presence of `NOT NULL` constraints under the "no information" interpretation [4]. In this context, Hartmann and Link established an equivalence of the implication problem for this class of FDs to that of propositional Horn clauses in Cadoli and Schaerf's family of $\mathcal{S}$-3 logics [15]. Both works only consider relations where FDs subsume UCs, but did not consider tables with duplicate rows. In this paper we cover the combined class of UCs and FDs in the presence of `NOT NULL` constraints under the possible world semantics of Codd's null "value unknown at present" in tables that can contain duplicate rows. No previous research has studied Armstrong relations in the presence of partial information and duplicate rows, never mind

NOT NULL constraints. This generality, however, is required for a more profound understanding of the semantics associated with SQL databases.

3 Codd Tables and Table Constraints

Codd's original proposal [7] to handle partial information suggested the addition to the database domains of an unmarked null value unk, whose meaning is "value unknown at present". Following Codd's proposal, partial information is represented in SQL by using unk as a distinguished null value [8].

Let $\mathfrak{H} = \{H_1, H_2, \ldots\}$ be a countably infinite set of symbols, called *column headers* or *headers* for short. A *table schema* is a finite non-empty subset T of $\mathfrak{H}$. Each header H of a table schema T is associated with an infinite domain $dom(H)$ of the possible values that can occur in column H. To encompass partial information every column may have an unmarked null value, denoted by $\texttt{unk} \in dom(H)$. The intention of unk is to mean "value unknown at present".

For header sets X and Y we may write XY for $X \cup Y$. If $X = \{H_1, \ldots, H_m\}$, then we may write $H_1 \cdots H_m$ for X. In particular, we may write simply H to represent the singleton $\{H\}$. A *row* over T (T-row or simply row, if T is understood) is a function $r : T \to \bigcup_{H \in T} dom(H)$ with $r(H) \in dom(H)$ for all $H \in T$. The null value occurrence $r(H) = \texttt{unk}$ associated with a header H in a row r means that the value $r(H)$ is unknown at present. For $X \subseteq T$ let $r[X]$ denote the restriction of the row r over T to X. A *table* t over T is a finite multiset of rows over T. Let r_1 and r_2 be two rows over T. It is said that r_1 *subsumes* r_2 if for every header $H \in T$, $r_1(H) = r_2(H)$ or $r_2(H) = \texttt{unk}$ holds.

For a row r over T and a set $X \subseteq T$, r is said to be X-total if for all $H \in X$, $r(H) \neq \texttt{unk}$. Similar, a table t over T is said to be X-total, if every row r of t is X-total. A table t over T is said to be a *total table* if it is T-total.

Extending ideas by Levene and Loizou [18] the set of all *possible worlds* relative to a table t over T, denoted by $Poss(t)$, is defined by

$$Poss(t) := \{t' \mid t' \text{ is a table over } T \text{ and there is a bijection } b : t \to t' \text{ such that } \forall r \in t, r \text{ is subsumed by } b(r) \text{ and } b(r) \text{ is } T\text{-total}\}.$$

This definition of possible worlds embodies the *closed world assumption* (CWA) [16], since only T-total rows from the table t can be present in $Poss(t)$.

A *uniqueness constraint* (UC) over a table schema T is a statement of the form $unique(X)$, where $X \subseteq T$. A table t over T is said to *satisfy* the UC $unique(X)$ over T, if there is some $t' \in Poss(t)$ such that for all $r_1, r_2 \in t'$, if $r_1 \neq r_2$, then $r_1[X] \neq r_2[X]$. The satisfaction of a UC in a table reduces to the satisfaction of a key when the table is T-total. In this case there is exactly one $t' \in Poss(t)$ and $\forall t' \in Poss(t)$ is equivalent to $\exists t' \in Poss(t)$.

A *functional dependency* (FD) over a table schema T is a statement of the form $X \to Y$, where $XY \subseteq T$. A table t over T is said to *satisfy* the FD $X \to Y$ over T, if there is some $t' \in Poss(t)$ such that for all $r_1, r_2 \in t'$, if $r_1[X] = r_2[X]$, then $r_1[Y] = r_2[Y]$. We note that the definition of satisfaction of an FD in a table reduces to the standard definition of the satisfaction of an FD when the table is

T-total. Finally we remark that the weak approach to satisfaction of an FD by a table allows a higher degree of uncertainty to be represented in the database than the strong approach (where an FD must be satisfied in all possible worlds) [18]. The disadvantage of the weak over the strong approach is that strongly satisfied FDs are easier to maintain [18]. Hence, both approaches complement one another. It is future work to combine strong and weak FDs.

Following Atzeni and Morfuni [4] a *null-free subdefinition* (NFS) over the table schema T is a an expression T_s where $T_s \subseteq T$. The NFS T_s over T is satisfied by a table t over T, denoted by $\models_t T_s$, if and only if t is T_s-total. SQL allows the specification of column headers as `NOT NULL`, cf. Example 1. NFSs occur in everyday database practice: the set of headers declared `NOT NULL` forms the single NFS over the underlying table schema.

We introduce an extension of the notion of *agree sets* of distinct rows to the presence of null values [5,19]. For two rows r_1, r_2 over table schema T we define

$$\begin{aligned} ag^s(r_1, r_2) &= \{H \in T \mid r_1(H) = r_2(H) \text{ and } r_1(H) \neq \texttt{unk} \neq r_2(H)\}, \\ ag^w(r_1, r_2) &= \{H \in T \mid r_1(H) = \texttt{unk} \text{ or } r_2(H) = \texttt{unk}\}, \\ ag(r_1, r_2) &= ag^s(t_1, t_2) \cup ag^w(t_1, t_2)\ . \end{aligned}$$

Intuitively, this definition makes perfect sense: i) two rows strongly agree on a column if they agree in all possible worlds, and ii) two rows weakly agree on a column if there is a possible world on which they agree on H. Next we establish a syntactic characterization of the satisfaction of UCs and FDs.

Proposition 1. *Let $XY \subseteq T$ and t be a table over T. Then*

1. *t satisfies unique(X) if and only if for all $r_1, r_2 \in t$, if $r_1 \neq r_2$, then $X \not\subseteq ag^s(r_1, r_2)$.*
2. *t satisfies $X \rightarrow Y$ if and only if for all $r_1, r_2 \in t$, if $X \subseteq ag^s(r_1, r_2)$, then $Y \subseteq ag(r_1, r_2)$.* □

Example 2. Consider the schema with column headers *Address*, *City* and *ZIP*. For the left table t the middle and right tables are elements of $Poss(t)$.

Address	*City*	*ZIP*
Pont Neuf	Paris	75001
Pont Neuf	`unk`	31000

Address	*City*	*ZIP*
Pont Neuf	Paris	75001
Pont Neuf	Paris	31000

Address	*City*	*ZIP*
Pont Neuf	Paris	75001
Pont Neuf	Toulouse	31000

Hence, t satisfies *unique*(*ZIP*), *unique*(*Address*,*City*), *Address* $\rightarrow$ *City* and *City* $\rightarrow$ *ZIP*. However, t violates *unique*(*Address*) and *Address* $\rightarrow$ *ZIP*. □

In schema design and maintenance data dependencies are normally specified as semantic constraints on the tables intended to be instances of the schema. During the design process or the lifetime of a database one usually needs to determine further dependencies which are implied by the given ones. Let T be a table schema, let $T_s \subseteq T$ denote an NFS over T, and let $\Sigma \cup \{\varphi\}$ be a set of UCs and FDs over T. We say that Σ *implies* φ in the presence of T_s, denoted by $\Sigma \models_{T_s} \varphi$, if every table t over T that satisfies Σ and T_s also satisfies φ. If Σ does not imply

φ in the presence of T_s we may also write $\Sigma \not\models_{T_s} \varphi$. Let $\Sigma^*_{T_s} = \{\varphi \mid \Sigma \models_{T_s} \varphi\}$ be the *semantic closure* of Σ. One can attempt to determine the semantic closure by a syntactic approach, e.g. by applying the inference rules from Table 1 below. These inference rules have the form

$$\frac{\text{premise}}{\text{conclusion}} \text{ condition},$$

and inference rules without any premises are called axioms. An inference rule is called *sound* for the implication of UCs and FDs in the presence of an NFS, if whenever the elements in the premise of the rule and the NFS are satisfied by some T-table and the elements and NFS satisfy the conditions of the rule, then the table also satisfies the element in the conclusion of the rule. For a finite set $\Sigma \cup \{\varphi\}$ of UCs and FDs and a set $\mathfrak{R}$ of inference rules let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote the *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\gamma = [\sigma_1, \ldots, \sigma_n]$ of UCs and FDs such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or results from an application of an inference rule in $\mathfrak{R}$ to some elements in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. For a finite set Σ of UCs and FDs, let $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ be its *syntactic closure* under inferences by $\mathfrak{R}$. A set $\mathfrak{R}$ of inference rules is said to be *sound* (*complete*) for the implication of UCs and FDs in the presence of an NFS if for every table schema T, for every NFS T_s over T and for every set Σ of UCs and FDs over T we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*_{T_s}$ ($\Sigma^*_{T_s} \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is said to be a (finite) *axiomatization* for the implication of UCs and FDs in the presence of an NFS if $\mathfrak{R}$ is both sound and complete.

4 The Implication of Constraints over Codd Tables

We establish a comprehensive analysis of the implication problem for UCs and FDs in the presence of an NFS. First, we characterize the implication problem by a finite ground axiomatization. Next, we develop an algorithm that decides the implication problem in time linear in the input. Finally, we establish an equivalence to the implication of goal and definite clauses in Cadoli and Schaerf's well-known family of $\mathcal{S}$-3 logics.

4.1 Axiomatic Characterization

Let $\mathfrak{S}$ denote the set of inference rules in Table 1.

Theorem 1. *The set $\mathfrak{S}$ is a finite axiomatization for the implication of UCs and FDs in the presence of an NFS.* □

Example 3. Consider the schema Contact with headers *Address*, *City* and *ZIP*, NFS $\text{Contact}_s = \{ZIP\}$, and

$$\Sigma = \{unqiue(Address, City), ZIP \rightarrow City\}.$$

Then $unique(Address, City, ZIP)$ can be inferred from Σ by $\mathfrak{S}$ by an application of the *null pullback rule* to $unique(Address, City)$ and the FD $Address, City, ZIP \rightarrow Address, City$. The latter can be inferred by a single application of the *reflexivity axiom*. □

Table 1. Axiomatization of UCs and FDs in the presence of an NFS

$\frac{unique(X)}{X \to Y}$ (demotion)	$\frac{}{XY \to X}$ (reflexivity)	$\frac{X \to YZ}{X \to Y}$ (decomposition)
$\frac{X \to Y \quad unique(Y)}{unique(X)} Y \subseteq XT_s$ (null pullback)	$\frac{X \to Y \quad Y \to Z}{X \to Z} Y \subseteq XT_s$ (null transitivity)	$\frac{X \to Y \quad X \to Z}{X \to YZ}$ (union)

4.2 Algorithmic Characterization

Data engineers do not always require the full semantic closure $\Sigma^*_{T_s}$ of a set Σ. Instead, they often need to decide if a given UC or FD φ is implied by Σ in the presence of T_s. It is usually inefficient to enumerate the elements of $\Sigma^+_{\mathfrak{S}}$ until φ is found, or all elements have been enumerated and φ is not among them. We will now establish an algorithm that decides in linear time in the input if $\Sigma \models_{T_s} \varphi$.

For a header set X of table schema T let $X^*_{\Sigma,T_s} := \{H \mid \Sigma \models_{T_s} X \to H\}$ denote the *header set closure* of X with respect to Σ and T_s. For a set $\Sigma \cup \{\varphi\}$ of UCs and FDs over T it suffices to compute the header set closure with respect to the set $\Sigma_{\mathrm{FD}} := \{X \to T \mid unique(X) \in \Sigma\} \cup \{X \to Y \mid X \to Y \in \Sigma\}$ and T_s.

Lemma 1. *Let Σ be a set of UCs and FDs over the table schema T with NFS T_s. Then the following holds:*

1. *$\Sigma \models_{T_s} X \to Y$ if and only if $\Sigma_{FD} \models_{T_s} X \to Y$, and*
2. *$\Sigma \models_{T_s} unique(X)$ if and only if $\Sigma_{FD} \models_{T_s} X \to T$ and there is some $unique(Z) \in \Sigma$ such that $Z \subseteq XT_s$.* □

However, the header set closure of a header set X can be computed as follows.

Algorithm 2 (NFSClosure(X,$\Sigma_{\mathbf{FD}}$,T_s,T))

Input: header set X, FD set Σ_{FD}, NFS T_s over table schema T
Output: header set closure $X^*_{\Sigma_{\mathrm{FD}},T_s}$ of X with respect to Σ_{FD} and T_s
Method:
(A0) CLOSURE := X;
(A1) `repeat`
  OLDCLOSURE := CLOSURE;
  `for all` $V \to W \in \Sigma_{\mathrm{FD}}$ `do`
    `if` $V \subseteq$ CLOSURE $\cap XT_s$ `then`
      CLOSURE := CLOSURE $\cup W$;
    `endif`;
  `enddo`;
`until` OLDCLOSURE = CLOSURE;
(A2) `return` CLOSURE; □

The size $|\varphi|$ of φ is the total number of headers occurring in φ, and the size $||\Sigma||$ of Σ is the sum of $|\sigma|$ over all elements $\sigma \in \Sigma$.

Theorem 3. *The problem whether a UC or FD φ is implied by a set Σ of UCs and FDs in the presence of an NFS T_s can be decided in $\mathcal{O}(||\Sigma \cup \{\varphi\}||)$ time.* □

Example 4. Consider the SQL table definition CONTACT with headers *Address*, *City* and *ZIP*, NFS $\text{CONTACT}_s = \{Address, ZIP\}$, and

$$\Sigma = \{unqiue(Address, City), ZIP \rightarrow City\}.$$

Then $\Sigma_{\text{FD}} = \{\{Address, City\} \rightarrow ZIP, ZIP \rightarrow City\}$ and the column header closure of $\{Address, ZIP\}$ with respect to Σ_{FD} and CONTACT_s is $\{Address, City, ZIP\}$. Since there is no $unique(Z) \in \Sigma$ such that $Z \subseteq \{Address, ZIP\}$ we conclude by Lemma 1 that the UC $unique(Address, ZIP)$ is not implied by Σ in the presence of CONTACT_s. □

4.3 Logical Characterization

Schaerf and Cadoli [20] introduced $\mathcal{S}$-3 logics as "a semantically well-founded logical framework for sound approximate reasoning, which is justifiable from the intuitive point of view, and to provide fast algorithms for dealing with it even when using expressive languages".

For a finite set $\mathcal{L}$ of propositional variables let $\mathcal{L}^\ell$ denote the set of all literals over $\mathcal{L}$, i.e., $\mathcal{L}^\ell = \mathcal{L} \cup \{\neg H' \mid H' \in \mathcal{L}\} \subseteq \mathcal{L}^*$ where $\mathcal{L}^*$ denotes the propositional language over $\mathcal{L}$. Let $\mathcal{S} \subseteq \mathcal{L}$. An $\mathcal{S}$-3 interpretation of $\mathcal{L}$ is a total function $\hat{\omega} : \mathcal{L}^\ell \rightarrow \{\mathbb{F}, \mathbb{T}\}$ that maps every variable $H' \in \mathcal{S}$ and its negation $\neg H'$ into opposite values ($\hat{\omega}(H') = \mathbb{T}$ if and only if $\hat{\omega}(\neg H') = \mathbb{F}$), and that does not map both a variable $H' \in \mathcal{L} - \mathcal{S}$ and its negation $\neg H'$ into $\mathbb{F}$ (we must not have $\hat{\omega}(H') = \mathbb{F} = \hat{\omega}(\neg H')$ for any $H' \in \mathcal{L} - \mathcal{S}$). An $\mathcal{S}$-3 interpretation $\hat{\omega} : \mathcal{L}^\ell \rightarrow \{\mathbb{F}, \mathbb{T}\}$ of $\mathcal{L}$ can be lifted to a total function $\hat{\Omega} : \mathcal{L}^* \rightarrow \{\mathbb{F}, \mathbb{T}\}$ by means of simple rules [20]. Since we are only interested in Horn clauses here we require the following two rules for assigning truth values to a Horn clause: (1) $\hat{\Omega}(\varphi') = \hat{\omega}(\varphi')$, if $\varphi' \in \mathcal{L}^\ell$, and (2) $\hat{\Omega}(\varphi' \vee \psi') = \mathbb{T}$ if and only if $\hat{\Omega}(\varphi') = \mathbb{T}$ or $\hat{\Omega}(\psi') = \mathbb{T}$. An $\mathcal{S}$-3 interpretation $\hat{\omega}$ is a *model* of a set Σ' of $\mathcal{L}$-formulae if and only if $\hat{\Omega}(\sigma') = \mathbb{T}$ holds for every $\sigma' \in \Sigma'$. We say that Σ' *$\mathcal{S}$-3 implies* an $\mathcal{L}$-formula φ', denoted by $\Sigma' \models^3_{\mathcal{S}} \varphi'$, if and only if every $\mathcal{S}$-3 interpretation that is a model of Σ' is also a model of φ'.

In a first step, we define the fragment of $\mathcal{L}$-formulae that corresponds to UCs and FDs in the presence of an NFS T_s over a table definition T. Let $\phi : T \rightarrow \mathcal{L}$ denote a bijection between T and the set $\mathcal{L} = \{H' \mid H \in T\}$ of propositional variables that corresponds to T. For an NFS T_s over T let $\mathcal{S} = \phi(T_s)$ be the set of propositional variables in $\mathcal{L}$ that corresponds to T_s. Hence, the variables in $\mathcal{S}$ are the images of those column headers of T declared `NOT NULL`.

We now extend ϕ to a mapping Φ from the set of UCs and FDs over T. For a UC $unique(H_1, \ldots, H_n)$ over T, let $\Phi(unique(H_1, \ldots, H_n))$ denote the goal clause $\neg H'_1 \vee \cdots \vee \neg H'_n$. For an FD $H_1, \ldots, H_n \rightarrow H$ over T, let $\Phi(H_1, \ldots, H_n \rightarrow H)$

denote the definite clause $\neg H'_1 \vee \cdots \vee \neg H'_n \vee H'$. For the sake of presentation, but without loss of generality, we assume that FDs have only a single column header on their right-hand side. As usual, disjunctions over zero disjuncts are interpreted as $\mathbb{F}$. In what follows, we may simply denote $\Phi(\varphi) = \varphi'$ and $\Phi(\Sigma) = \{\sigma' \mid \sigma \in \Sigma\} = \Sigma'$.

Our aim is to show that for every SQL table definition T, for every set $\Sigma \cup \{\varphi\}$ of UCs and FDs and for every NFS T_s over T, there is some T_s-total table t that satisfies Σ and violates φ if and only if there is an $\mathcal{S}$-3 model $\hat{\omega}_t$ of Σ' that is not an $\mathcal{S}$-3 model of φ'. For arbitrary tables t it is not obvious how to define the $\mathcal{S}$-3 interpretation $\hat{\omega}_t$.

However, for deciding the implication problem $\Sigma \models_{T_s} \varphi$ it suffices to examine two-row tables (instead of arbitrary tables). For two-row tables $\{r_1, r_2\}$ we define the *special-3-interpretation* of $\mathcal{L}$ by

- $\hat{\omega}_{\{r_1,r_2\}}(H') = \mathbb{T}$ and $\hat{\omega}_{\{r_1,r_2\}}(\neg H') = \mathbb{F}$, if $\mathtt{unk} \neq r_1(H) = r_2(H) \neq \mathtt{unk}$,
- $\hat{\omega}_{\{r_1,r_2\}}(H') = \mathbb{T}$ and $\hat{\omega}_{\{r_1,r_2\}}(\neg H') = \mathbb{T}$, if $r_1(H) = \mathtt{unk}$ or $r_2(H) = \mathtt{unk}$,
- $\hat{\omega}'_{\{r_1,r_2\}}(H') = \mathbb{F}$ and $\hat{\omega}'_{\{r_1,r_2\}}(\neg H') = \mathbb{T}$, if $\mathtt{unk} \neq r_1(H) \neq r_2(H) \neq \mathtt{unk}$

for all $H' \in \mathcal{L}$. In particular, if $\{r_1, r_2\}$ is T_s-total, then $\hat{\omega}_{\{r_1,r_2\}}$ is an $\mathcal{S}$-3 interpretation.

Theorem 4. *Let $\Sigma \cup \{\varphi\}$ be a set of UCs and FDs over the SQL table definition T, and let T_s denote an NFS over T. Let $\mathcal{L}$ denote the set of propositional variables that corresponds to T, $\mathcal{S}$ the set of variables that corresponds to T_s, and $\Sigma' \cup \{\varphi'\}$ the set of goal and definite clauses over $\mathcal{L}$ that corresponds to $\Sigma \cup \{\varphi\}$. Then $\Sigma \models_{T_s} \varphi$ if and only if $\Sigma' \models^3_{\mathcal{S}} \varphi'$.* □

Example 5. Consider the table schema CONTACT with headers *Address*, *City* and *ZIP*, NFS $\text{CONTACT}_s = \{Address, ZIP\}$, and

$$\Sigma = \{unqiue(Address, City), ZIP \rightarrow City\}.$$

Then the UC $unique(Address, ZIP)$ is not implied by Σ in the presence of CONTACT_s, as the following SQL table t demonstrates:

Address	*City*	*ZIP*
Pont Neuf	Toulouse	31000
Pont Neuf	unk	31000

.

Indeed, the special $\mathcal{S}$-3 interpretation $\hat{\omega}_t$ where for all $L \in \mathcal{L}^\ell$, $\hat{\omega}_t(L) = \mathbb{F}$ iff $L \in \{\neg Address', \neg ZIP'\}$ is an $\mathcal{S}$-3 model of Σ' but not a model of φ'_2. □

5 Armstrong Tables

In this section we extend Demetrovics, Mannila, Räihä, Beeri, Dowd, Fagin and Statman's results on the structural and computational properties of Armstrong relations for FDs from total relations [5,10,19] to the combined class of UCs and FDs in the presence of an NFS over Codd tables.

5.1 Characterization

First we would like to establish sufficient and necessary conditions when a given table is an Armstrong table with respect to a given set Σ of UCs and FDs and an NFS T_s. This would generalize a well-known result by Mannila, Räihä, Beeri, Dowd, Fagin and Statman for FDs over total database relations [5,19]. Especially useful in this regard is Mannila and Räihä's notion of maximal sets [19] which we generalize here from total relations to Codd tables.

Definition 1. *Let Σ be a set of UCs and FDs and let T_s be an NFS over table schema T. For a column header $H \in T$ we define the maximal sets $max_{\Sigma,T_s}(H)$ of H with respect to Σ and T_s as follows:*

$$max_{\Sigma,T_s}(H) := \{X \subseteq T \mid \Sigma \not\models_{T_s} X \to H \wedge \forall H' \in T - X(\Sigma \models_{T_s} XH' \to H)\}.$$

The maximal sets *of T with respect to Σ and T_s are defined as $max_{\Sigma,T_s}(T) = \bigcup_{H \in T} max_{\Sigma,T_s}(H)$. If Σ and T_s are clear from the context we may simply write $max(H)$ and $max(T)$, respectively.* □

Thus, the maximal sets of a column header H with respect to Σ and T_s are the maximal header subsets of T that do not functionally determine H.

Example 6. Consider the table schema Contact with headers *Address*, *City* and *ZIP*, NFS $T_s = \text{Contact}_s = \{ZIP\}$, and

$$\Sigma = \{unqiue(Address, City), ZIP \to City\}.$$

Then the maximal sets for the column headers are:

- $max_{\Sigma,T_s}(Address) = \{\{City, ZIP\}\}$,
- $max_{\Sigma,T_s}(City) = \{\{Address\}\}$,
- $max_{\Sigma,T_s}(ZIP) = \{\{Address\}, \{City\}\}$. □

The idea is that it is a necessary condition for an Armstrong table that for each maximal set there must be distinct rows in the table whose strong agree set is the maximal set. This is to guarantee that all the FDs not implied by the set of UCs and FDs in the presence of an NFS are violated. Over tables, however, it is still possible that there are UCs $unique(X)$ not implied by Σ in the presence of T_s over T, even if the FD $X \to T$ is implied. For this reason we also require of Armstrong tables that for all column header sets X that are maximal with this property there must be distinct rows in the table whose strong agree set is X. This motivates the following definition.

Definition 2. *Let Σ be a set of UCs and FDs and let T_s be an NFS over table schema T. We define the* duplicate sets *$dup_{\Sigma,T_s}(T)$ of T with respect to Σ and T_s as follows:*

$$dup_{\Sigma,T_s}(T) := \{X \subseteq T \mid \Sigma \models_{T_s} X \to T \wedge \Sigma \not\models_{T_s} unique(X) \wedge \forall H' \in T - X(\Sigma \models_{T_s} unique(XH'))\}.$$

If Σ and T_s are clear from the context we may simply write $dup(T)$. □

Example 7. Consider the table schema CONTACT with headers *Address*, *City* and *ZIP*, NFS $T_s = \text{CONTACT}_s = \{ZIP\}$, and

$$\Sigma = \{unqiue(Address, City), ZIP \rightarrow City\}.$$

Then we have $dup_{\Sigma,T_s}(\text{CONTACT}) = \{\{Address, ZIP\}\}$. □

For our anticipated characterization of Armstrong tables the notion of a (strong) agree set plays an important role. While strong and weak agree sets coincide over total relations, the distinction between the two is crucial for Codd tables. Indeed, we require an additional notion that helps us to ensure that i) for each maximal set of a column header there are rows that strongly agree on the maximal set but disagree on the column header, and ii) each strong agree set includes all column headers functionally determined by it.

Definition 3. *Let T be a table schema, and t a table over T. For $X \in ag^s(r)$ let $w(X) = \bigcap\{Y \mid \exists r, r' \in t(X = ag^s(r, r') \wedge Y = ag(r, r'))\}$.* □

Example 8. Let t denote the Codd table from Example 1, i.e. the table on the right. Here we obtain

- $ag^s(t) = \{\{City, ZIP\}, \{Address\}, \{City\}, \{Address, ZIP\}\}$,
- $ag^w(t) = \{\{Address\}, \{City\}\}$, and
- $w(Address) = \{Address\}$, $w(City) = \{City\}$, $w(City, ZIP) = \{City, ZIP\}$, and $w(Address, ZIP) = \{Address, City, ZIP\}$. □

These notions allow us to obtain the following characterization of Armstrong tables for a given Codd table.

Theorem 5. *Let T be a table schema, Σ a set of UCs and FDs, and T_s an NFS over T. For all tables t over T it holds that t is an Armstrong table for Σ and T_s if and only if all of the following conditions are satisfied:*

1. $\forall H \in T \forall X \in max_{\Sigma,T_s}(H)(X \in ag^s(t) \wedge H \notin w(X))$,
2. $\forall X \in ag^s(t)(X^*_{\Sigma,T_s} \subseteq w(X))$,
3. $\forall X \in dup_{\Sigma,T_s}(T)(X \in ag^s(t))$,
4. $\forall X \in ag^s(t) \forall unique(Z) \in \Sigma(Z \not\subseteq X)$,
5. $total(t) = T_s$. □

Example 9. Consider again the table schema CONTACT with headers *Address*, *City* and *ZIP*, NFS $T_s = \text{CONTACT}_s = \{ZIP\}$, and

$$\Sigma = \{unqiue(Address, City), ZIP \rightarrow City\}.$$

Examples 6, 7 and 8 allow us to verify all the conditions of Theorem 5 for the Codd table t on the right of Example 1. Hence, t is indeed an Armstrong table for Σ and T_s. □

5.2 Computation

In this section we establish an algorithm that computes for any given table schema T, any given set Σ of UCs and FDs, and any given NFS T_s over T an Armstrong table for Σ and T_s. Following Lemma 1 and Theorem 5 we aim to compute the maximal set families $max_{\Sigma_{\mathrm{FD}},T_s}(T)$ and the duplicate sets $dup_{\Sigma,T_s}(T)$.

Lemma 2. *Let T be a table schema, T_s an NFS over T, and $\Sigma_{FD} = \Sigma' \cup \{X \to A\}$ a set of FDs over T. For $WC \subseteq T$, it takes $\mathcal{O}(|T| \times ||\Sigma||)$ time to test whether $W \in max_{\Sigma_{FD},T_s}(C)$.* □

The maximal sets for T with respect to Σ_{FD} and T_s can be computed by testing all subsets of T. This, however, will hardly be efficient. The following result establishes an iterative approach for computing the maximal sets for T with respect to Σ_{FD} and T_s. The algorithm starts with the maximal sets for T with respect to an empty FD set in the presence of T_s, and then adds the FDs of Σ_{FD} one by one while monitoring the resulting changes to the family of maximal sets.

Theorem 6. *Let T be a table schema, T_s an NFS over T, and $\Sigma_{FD} = \Sigma'_{FD} \cup \{X \to A\}$ a set of FDs over T. For $H \in T$ let $V \in max_{\Sigma_{FD},T_s}(H)$. Then $V \in max_{\Sigma'_{FD},T_s}(H)$ or ($H = A$ or $A \in T_s$) holds and there is some $H' \in X - V$ such that*

i) $VH' \in max_{\Sigma'_{FD},T_s}(H)$, if $X \not\subseteq T_s$, or
ii) $V = W \cap Z$ for some $W \in max_{\Sigma'_{FD},T_s}(H)$ and some $Z \in max_{\Sigma'_{FD},T_s}(H')$. □

By Theorem 6, the family $max_{\Sigma_{\mathrm{FD}},T_s}(H)$ can be computed from the family $max_{\Sigma'_{\mathrm{FD}},T_s}(H)$ as follows. For each $V \in max_{\Sigma'_{\mathrm{FD}},T_s}(H)$ such that $V \notin max_{\Sigma_{\mathrm{FD}},T_s}(H)$, if $X = \emptyset$, then $max_{\Sigma_{\mathrm{FD}},T_s}(H) = \emptyset$, otherwise, for each $H' \in X$, if $H' \notin T_s$, then $V - H' \in max_{\Sigma_{\mathrm{FD}},T_s}(H)$, and for each $Z \in max_{\Sigma'_{\mathrm{FD}},T_s}(H')$ test if $V \cap Z \in max_{\Sigma_{\mathrm{FD}},T_s}(H)$.

To compute $dup_{\Sigma,T_s}(T)$ we generate the hyper-graph $\mathcal{H} = (V, E)$ with vertex set $V = T$ and the set $E = \{X - T_s \mid unique(X) \in \Sigma_{\mathrm{UC}}\}$ as hyper-edges. From this we obtain $dup_{\Sigma,T_s}(T)$ as

$$dup_{\Sigma,T_s}(T) = \{T - X \mid X \in Tr(\mathcal{H}) \ \wedge \ \forall M \in max_{\Sigma_{\mathrm{FD}},T_s}(T)(T - X \not\subseteq M)\}$$

where $Tr(\mathcal{H})$ denotes the *minimal transversals* of the hyper-graph $\mathcal{H}$ [14].

The following algorithm computes an Armstrong table for an arbitrary set Σ of UCs and FDs and an arbitary NFS T_s. In step (A0) the algorithm utilizes Theorem 6 to compute the family of maximal sets with respect to Σ_{FD}. In steps (A4)-(A8) the strong agree sets required for a Codd table to be Armstrong according to Theorem 5 are generated with respect to the row r_0. Due to some non-standard UCs and FDs some columns might have a constant non-null entry. Finally, an additional row may be required to introduce the null value `unk` to columns that are specified `NULL`, see step (A9).

Algorithm 7 (Armstrong table computation)

Input: table schema T, a set Σ of UCs and FDs and an NFS T_s over T
Output: Armstrong table t for Σ and T_s
Method: let $c_{H,0}, c_{H,1}, \ldots \in dom(H)$ be distinct
(A0) `for all` $H \in T$ compute $max_{\Sigma_{\mathrm{FD}},T_s}(H)$;
(A1) $t := \{r_0\}$ where `for all` $H \in T$, $r_0(H) := c_{H,0}$;
(A2) $Const := \emptyset^*_{\Sigma,T_s}$;
(A3) $i := 1$;
(A4) `for all` $X \in max_{\Sigma_{\mathrm{FD}},T_s}(T) \cup dup_{\Sigma,T_s}(T)$ `do`
(A5) `if` $X \in max_{\Sigma_{\mathrm{FD}},T_s}(T)$, `then` $Z := \{H \in T \mid X \in max_{\Sigma_{\mathrm{FD}},T_s}(H)\}$;
`else` $Z := \emptyset$;
`endif`;
(A6) $t := t \cup \{r_i\}$ where for all $H \in T$,

$$r_i(H) := \begin{cases} c_{H,0} & \text{, if } H \in X \cup (Const \cap T_s) \\ c_{H,i} & \text{, if } H \in Z \cup (T_s - (X \cup Const)) \\ \texttt{unk} & \text{, else} \end{cases};$$

(A7) $i := i + 1$;
(A8) `enddo`;
(A9) $total(t) := \{H \in T \mid \forall r \in t(r[H] \neq \texttt{unk})\}$;
`if` $total(t) - T_s \neq \emptyset$,
`then return` $t := t \cup \{r_i\}$ where for all $H \in T$,

$$r_i(H) := \begin{cases} \texttt{unk} & \text{, if } H \in (total(t) \cup Const) - T_s \\ c_{H,0} & \text{, if } H \in Const \cap T_s \\ c_{H,i} & \text{, else} \end{cases};$$

`else return` r;
`endif`; □

The correctness of Algorithm 7 follows essentially from Theorems 5 and 6.

Theorem 8. *Algorithm 7 computes an Armstrong table for Σ and T_s.* □

Example 10. Consider again the table schema CONTACT with headers *Address*, *City* and *ZIP*, NFS $T_s = \text{CONTACT}_s = \{ZIP\}$, and

$$\Sigma = \{unqiue(Address, City), ZIP \rightarrow City\}.$$

Examples 6 and 7 show the families of maximal and duplicate sets for Σ and T_s. Algorithm 2 would compute the following Armstrong table for Σ and T_s.

Address	*City*	*ZIP*
$c_{A,0}$	$c_{C,0}$	$c_{Z,0}$
$c_{A,1}$	$c_{C,0}$	$c_{Z,0}$
$c_{A,0}$	$c_{C,2}$	$c_{Z,2}$
unk	$c_{C,0}$	$c_{Z,3}$
$c_{A,0}$	unk	$c_{Z,0}$

Address	*City*	*ZIP*
Pont Neuf	Paris	75001
Le Louvre	Paris	75001
Pont Neuf	Toulouse	31000
unk	Paris	75007
Pont Neuf	unk	75001

A suitable substitution results in the Armstrong table from Example 1. □

We present some results regarding the time and space complexity required for the representation of constraints using Codd tables.

The Time-Complexity to Find Armstrong Tables. We recall what we mean by *precisely exponential* [5]. Firstly, it means that there is an algorithm for computing an Armstrong table, given a set Σ of UCs and FDs and an NFS T_s, where the running time of the algorithm is exponential in the number of attributes. Secondly, it means that there is a set Σ of UCs and FDs and an NFS T_s in which the number of rows in each minimum-sized Armstrong table for Σ and T_s is exponential — thus, an exponential amount of time is required in this case simply to write down the relation.

Proposition 2. *The complexity of finding an Armstrong table, given a set of UCs and FDs and an NFS, is precisely exponential.* □

The Size of our Computed Armstrong Tables. It is a practical question to ask how many rows a minimum-sized Armstrong table requires. An Armstrong table t for Σ and T_s is said to be *minimum-sized* if there is no Armstrong table t' for Σ and T_s such that $|t'| < |t|$.

Theorem 9. *On input (T, Σ, T_s), Algorithm 7 computes an Armstrong table for Σ and T_s whose size is at most quadratic in the size of a minimum-sized Armstrong table for Σ and T_s.* □

The Size of Representations. None of the representations strictly dominates the other. Therefore, both representations should be used together.

Theorem 10. *There is some table schema T, some set Σ of UCs and FDs and some NFS T_s over T such that Σ has size $\mathcal{O}(n)$, and the size of a minimum-sized Armstrong table for Σ and T_s is $\mathcal{O}(2^{n/2})$. There is some table schema T, some set Σ of UCs and FDs and some NFS T_s over T such that there is an Armstrong table for Σ and T_s where the number of rows is in $\mathcal{O}(n)$, and the optimal cover of Σ with respect to T_s has size $\mathcal{O}(2^n)$.* □

Constraint sets can help to identify constraints incorrectly perceived as meaningful, and Armstrong tables can help to identify constraints incorrectly perceived as meaningless [17].

6 Conclusion

We investigated the combined class of UCs and FDs over Codd tables under a weak possible world semantics. We characterized the implication problem axiomatically, algorithmically and logically. Our results on the representation of constraint sets as Codd tables subsume classical findings. Our results show that there is no penalty in generalizing the theory for total relations to Codd tables that occur in real database systems. For the future we plan to implement our results in a design aid, and to consider the combination of both weak and strong possible world semantics [18].

Acknowledgement. This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand.

The second author is supported by a research grant of the Alfried Krupp von Bohlen and Halbach foundation, administered by the German Scholars organization.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Alexe, B., Kolaitis, P., Tan, W.-C.: Characterizing schema mappings via data examples. In: Proceedings to the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 261–271 (2010)
3. Armstrong, W.W.: Dependency structures of database relationships. Information Processing 74, 580–583 (1974)
4. Atzeni, P., Morfuni, N.: Functional dependencies and constraints on null values in database relations. Information and Control 70(1), 1–31 (1986)
5. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of Armstrong relations for functional dependencies. J. ACM 31(1), 30–46 (1984)
6. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (1970)
7. Codd, E.F.: Extending the database relational model to capture more meaning. ACM Trans. Database Syst. 4(4), 397–434 (1979)
8. Date, C., Darwen, H.: A guide to the SQL standard. Addison-Wesley Professional, Reading (1997)
9. De Marchi, F., Petit, J.-M.: Semantic sampling of existing databases through informative Armstrong databases. Inf. Syst. 32(3), 446–457 (2007)
10. Demetrovics, J.: On the equivalence of candidate keys with Sperner systems. Acta Cybern. 4, 247–252 (1980)
11. Diederich, J., Milton, J.: New methods and fast algorithms for database normalization. ACM Trans. Database Syst. 13(3), 339–365 (1988)
12. Fagin, R.: Functional dependencies in a relational data base and propositional logic. IBM Journal of Research and Development 21(6), 543–544 (1977)
13. Fagin, R.: Armstrong databases. Technical Report RJ3440(40926), IBM Research Laboratory, San Jose, California, USA (1982)
14. Gottlob, G., Pichler, R., Wei, F.: Tractable database design through bounded treewidth. Inf. Syst. 35(3), 278–298 (2010)
15. Hartmann, S., Link, S.: When data dependencies over SQL tables meet the Logics of Paradox and *S*-3. In: PODS Conference (2010)
16. Imielinski, T., Lipski Jr., W.: Incomplete information in relational databases. J. ACM 31(4), 761–791 (1984)
17. Langeveldt, W.-D., Link, S.: Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies. Inf. Syst. 35(3), 352–374 (2010)
18. Levene, M., Loizou, G.: Axiomatisation of functional dependencies in incomplete relations. Theor. Comput. Sci. 206(1-2), 283–300 (1998)
19. Mannila, H., Räihä, K.-J.: Design by example: An application of Armstrong relations. J. Comput. Syst. Sci. 33(2), 126–141 (1986)
20. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artif. Intell. 74, 249–310 (1995)
21. Thalheim, B.: Entity-Relationship modeling. Springer, Heidelberg (2000)

Efficient Early Top-k Query Processing in Overloaded P2P Systems

William Kokou Dédzoé[1], Philippe Lamarre[1], Reza Akbarinia[2], and Patrick Valduriez[2]

[1] LINA, University of Nantes, France
{William.Dedzoe,Philippe.Lamarre}@univ-nantes.fr
[2] INRIA and LIRMM, Montpellier, France
{Reza.Akbarinia,Patrick.Valduriez}@inria.fr

Abstract. Top-k query processing in P2P systems has focused on efficiently computing the top-k results while reducing network traffic and query response time. However, in overloaded P2P systems (with very high query loads), some peers may take a long time to answer, thus making the user wait a long time to obtain the final top-k result. In this paper, we address this problem, which we reformulate as early top-k query processing in P2P systems. First, to complement response time, we introduce two new metrics, stabilization time and cumulative quality gap, with which we formally define the problem. Then, we propose an efficient algorithm that dynamically adapts to query loads of peers in order to return to the user top-k results as soon as possible, without waiting for the final result. We validated our solution through simulations over a real dataset. The results show that our algorithm significantly outperforms baseline algorithms by returning high quality top-k results to users in much better times.

1 Introduction

Top-k query processing in Peer-to-Peer (P2P) systems has received a lot of attention [6,18,19,1,2]. The main reason for such interest is that they reduce the network traffic and avoid overwhelming the user with large numbers of uninteresting answers. With a top-k query, the user specifies a number k of the most relevant answers to be returned by the system. The quality (i.e. score of relevance) of the answers to the query is determined by user-specified scoring functions [9].

Despite the fact that these top-k query processing solutions reduce network traffic, they may significantly delay the answers to users. This is because top-k results are usually returned to the user only when all queried peers have finished processing the query. Thus, query response time is dominated by the slowest queried peer, which makes users suffer from long waiting times. Indeed, this becomes even more problematic when peers are overloaded, i.e. in overloaded P2P systems. Therefore, current top-k processing solutions (e.g. [1] and [6]) are not suitable for many popular P2P applications, such as P2P web search engines

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 140–155, 2011.

and P2P data sharing for online communities, because they are often exposed to a large number of incoming queries and thus may easily become overloaded.

In this paper, we address the problem of reducing users waiting time when performing top-k query processing in the context of overloaded P2P systems. We reformulate this problem as *early top-k query processing in P2P systems*. We revisit top-k query processing by considering two new metrics to complement *response time*: *stabilization time* and *cumulative quality gap*. Then, to cope with this problem, we propose an algorithm that dynamically adapts to query loads of peers so as to return to users top-k results as soon as possible, without waiting for the final results. To the best of our knowledge, this is the first work that deals with this problem.

In summary, we make the following contributions in this paper:

- We formally define the problem of early top-k query processing in P2P systems using both stabilization time and cumulative quality gap.
- We propose QUAT[1], an efficient algorithm for early top-k query processing. In QUAT, each peer maintains a description of its local data and the descriptions of its neighborhood (i.e. the descriptions of data owned locally by its direct neighbors and data owned locally by these neighbors direct neighbors). These descriptions allow peers to prioritize the queries that can provide high quality results, and to forward them in priority to the neighbors that can provide high quality answers.
- We validate our solution through a thorough experimental evaluation using a real-world dataset. The results show that QUAT significantly outperforms baseline algorithms by returning faster the final top-k results to users. They also demonstrate that in the presence of peer failures, QUAT provides top-k results with good accuracy compared to baseline algorithms.

The rest of this paper is organized as follows. In section 2, we make precise the P2P system model that we consider, with basic definitions regarding top-k queries. Section 3 defines the early top-k query processing problem. In Section 4, we present the QUAT algorithm. Section 5 presents how peers build and maintain routing indices based on their local and neighbors descriptions for top-k query processing. In Section 6, we give our performance evaluation of QUAT. Section 7 discusses related work. In Section 8, we conclude.

2 P2P System Model

In this section, we first describe the general model of unstructured P2P which we consider for describing our solution[2]. Then, we provide a model and base definitions for top-k queries.

[1] Quality-based Early Top-k Query Processing refers to Khat, an African plant whose leaves are chewed as a stimulant.

[2] It is worth noting that our solution can be easily adapted to structured P2P systems as well.

2.1 System Model

We model an unstructured P2P network of n peers as an undirected graph $G = (P, E)$, where $P = \{p_0, p_1, \cdots, p_{n-1}\}$ is the set of peers and E the set of connections between the peers. We denote by $N(p_i)$, the set of peers to which p_i is directly connected, so $N(p_i) = \{p_j | (p_i, p_j) \in E\}$. The value $\|N(p_i)\|$ is called the degree of p_i. The average degree of peers in G is called the *average degree* of G and is denoted by φ. In our model, we assume horizontal data distribution to the n peers. Each peer $p \in P$ holds and maintains a set $D(p)$ of data items such as relational data (i.e. tuples).

Let c_i be the number of queries which a peer p_i can process per time unit. We call c_i the capacity of p_i. If a peer receives queries from its neighbors at a rate higher than its capacity c_i, then the queries are queued until the receiving peer processes these queries. Note that the maximal number of connections (communication channels) which a peer can open simultaneously with its neighbors is proportional to the capacity of the peer. However, peers may set this number lower than the maximal value if they wish to.

2.2 Top-k Queries

We model each top-k query q by a tuple $< qid, \bar{q}, ttl, k, f, p_0 >$ such that qid is the query identifier, $\bar{q}$ is the query itself (e.g. SQL query), $ttl \in \mathbb{N}$ (Time-To-Live) is the maximum hop distance set by the user, $k \in \mathbb{N}^*$ is the number of results requested by the user, f is a scoring function that denotes the score of relevance (i.e. the quality) of a given data item to a given query and $p_0 \in P$ the originator of query q. We assume that the data items scores are in $[0, 1]$. A top-k result set of a given query q is the k top results among data items owned by all peers that receive q. The data item in top-k result set having the lowest score is called the $mink$ of that top-k result set.

In our system, a query is forwarded from the query originator to its neighbors until the Time-To-Live value of the query decreases to 0 or the current peer has no peer to forward the query. So the query processing flow can be represented as a tree, which is called the query forwarding tree.

2.3 Peer Description

In our system, each peer is described by a synthetic description based on the data items owned by the peer. The approach of building this synthetic synthetic description is out of the scope of this paper. We assume that it is obtained through a description aggregation function which takes as input a set of data items and generates a single description of these data items e.g. [8] and [16]. We make the following assumptions regarding the description aggregation function:

- It is incremental, i.e. a peer that adds or removes a data item does not cause a total reconstruction of its description.
- It is composable, i.e. is possible to create a single description using two or more descriptions.

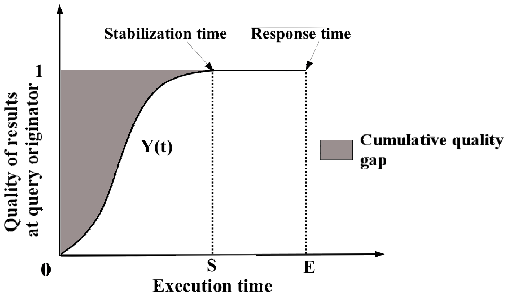


Fig. 1. Quality of top-k results at the query originator wrt. execution time

– It is optimistic, i.e. the estimation of a top-k query's result quality with respect to description should not be lower than the exact scores of data (i.e. data which are used to build this semantic description).

Notice that, there exist descriptions that satisfy these assumptions in the literature, e.g. semantic descriptions [16].

3 Problem Definition

Let us first give our assumptions regarding schema management and the unstructured P2P architecture. We assume that peers are able to express queries over their own schema without relying on a centralized global schema. Several solutions have been proposed to support decentralized schema mapping and we simply assume it is provided using one of the existing techniques, *e.g.* [12]. In the following, we first give some definitions and formally state the problem.

3.1 Preliminaries

To process a top-k query in a P2P system, our approach provides intermediate results to users as soon as peers process the query locally. This allows users to progressively see the evolution of their query execution by receiving intermediate results. Notice that at some point of query execution, the top-k intermediate results received by the user may not change any more, because the user has already received all top-k results. We denote this point as the **stabilization time** (see Figure 1). The stabilization time may be much lower than the response time (when there is no more top-k result).

Recall that our goal is to return high-quality results to the user as soon as possible. To capture this, we introduce the *quality evolution* concept as follows. Given a top-k query q, we define the quality evolution $Y(t)$ of q at time t as the sum of scores of q's intermediate top-k results at t and q's originator. To be independent of the scoring values (which can be different from one query to another), we normalize the quality evolution of a query. With this in mind, we

divide the quality evolution of a given query by the sum of scores of the final top-k results of that query. Thus, the quality evolution values are in the interval $[0, 1]$ and the quality of the top-k final results is equal to 1.

The quality of intermediate top-k results at the query originator evolves during query execution. Let us now introduce the **cumulative quality gap**, which is the sum of the quality difference between intermediate top-k result sets received until the stabilization time and the final top-k result set (see Figure 1). Notice that the smaller is the cumulative gap the higher is the quality of intermediate results returned to the user. We formally define the cumulative gap as follows.

Definition 1 ***Cumulative quality gap.*** *Let q be a top-k query, $Y(t)$ the quality evolution of q at time t at the query originator and s be the stabilization time of q. The cumulative quality gap of the query q, denoted by c_{qg} is:*

$$c_{qg} = \int_0^s (1 - Y(t))\,\mathrm{d}t = s - \int_0^s Y(t)\,\mathrm{d}t \tag{1}$$

In this paper, we address top-k query processing in overloaded P2P systems wherein peers might receive many queries in a short period of time. For this we define *stabilization time* and *cumulative quality gap* for *time periods*, and our objective is to develop algorithms that are efficient in terms of them. The *stabilization time* and *cumulative quality for time periods* T are respectively the average of the stabilization and the average of the cumulative quality gap for the queries issued in T.

Notice that, one can consider the precision of intermediate top-k results as a metric to characterize early top-k algorithms. However, the precision does not reflect if users receive high quality results early.

3.2 Problem Statement

Given a time period T, let S_T and C_{qgT} be the stabilization time and cumulative quality gap over T respectively. Our goal is to reduce S_T and C_{qgT} while providing the correct top-k result sets.

4 Quat Top-k Query Processing

In QUAT, each peer maintains a description of its local data and the descriptions of its neighborhood (i.e. the descriptions of data owned locally by its direct neighbors and data owned locally by these neighbors direct neighbors). These descriptions are used to create routing indices for top-k query processing. We give more details on the construction and maintenance of these routing indices in Section 5. Top-k query processing in QUAT proceeds in following phases: 1) query initialisation; 2) query forwarding; 3) local execution of the query by peers; 4) bubbling up of the peers results for the query along the query forwarding tree.

4.1 Query Initialisation

Query processing starts at the query originator, i.e. the peer at which a user issues a top-k query q. Note that the scoring function f and the number of results k wished by the user are specified in q. The query originator performs some initializations. First, it sets ttl which is either user-specified or default. Second, it creates a unique identifier qid for q which is useful to distinguish between new queries and those received before. qid is made of a unique peer identifier and a query counter managed by the query originator. Then, q is included in a message that is broadcast by the query originator to its reachable neighbors.

4.2 Query Forwarding

In classical query forwarding approaches [1,6], a peer forwards any incoming query to all its neighbors in parallel. However, in overloaded P2P systems, this approach may quickly collapse the system as it usually demands a lot of computing resources. Thus, we consider that a peer forwards a query to at most m neighbors in parallel, where m depends on the current query load of the peer. When m is smaller than the total number of neighbors of a peer, the peer must decide in which order to forward the query to its neighbors. To do so, the peer sorts its neighbors based on its neighborhood's data descriptions, i.e. by estimating the results quality which each neighbor can provide for the query. Indeed, when all m connections are allocated, the peer sends the query to another neighbor as a connection gets released. Notice that, each peer includes its current top-k intermediate results into their query messages in order to avoid a peer sending less interesting results than those computed so far. Note that the top-k intermediate result set at given peer is the k best results of both the results the peer received so far from its neighbors and its local results (if any). Furthermore, these top-k intermediate results are also used by peers to avoid sending the query to those neighbors that cannot return results better than the $mink$ of these top-k intermediate results, i.e. by using the neighborhood's data descriptions. Notice that, we use the query load of each peer to set its value of m. However, it is possible to take also into account loads of neighbors of peers and the overall load of the system if they can be obtained.

4.3 Local Query Execution

In current approaches [1,6], a peer executes incoming queries as they arrive, i.e. using a *First-In-First-Out policy.* However, in overloaded P2P systems, wherein query queues at peers are often very long, this approach can significantly increase users waiting times. This is because queries for which peers can provide results of high scores may be penalized for those they can only provide results of low scores. To cope with this problem, the order in which incoming queries are executed locally by peers depends on the estimation of their results quality with respect to their local data descriptions. Due to the fact that peers forward in priority queries having best results quality estimation, peers may receive from their neighbors,

results for queries which they have not yet executed locally. In this case, the optimistic property of peers' data descriptions allows peers to avoid executing queries which they cannot provide results whose scores will be better than that of the *mink* of their current top-k intermediate results.

4.4 Bubbling Up Results

A naive solution to reduce the user waiting time is to return the top-k results from the peers directly to the query originator as soon as they have done executing the query. However, returning high numbers of results increases network traffic and can quickly cause a bottleneck at the query originator. For this in QUAT, when a peer submits a top-k query q, the local results of the peers that have received q are bubbled up to the query originator using query q's forwarding tree. The technique of bubble up results of peers using query forwarding tree is very interesting because peers can use intermediate results received from its children to avoid executing locally some queries.

In QUAT, a peer's decision to send intermediate results is based on the improvement impact brought by its current top-k intermediate result set over the top-k intermediate result set it has already sent to its parent. This improvement impact can be computed by using the score of top-k results in the result set. Therefore, we introduce the notion of score-based improvement impact. Intuitively, the score-based improvement impact at a given peer for a given top-k query is the gain of score of peer's current top-k intermediate set compared to the top-k intermediate set it sent so far.

Definition 2 ***Score-based improvement impact.*** *Given a top-k query q, and peer $p \in \bar{P}$ (where $\bar{P}$ is the set of peers which received q), let T_{cur} be the current top-k intermediate set of q at p and T_{old} be the top-k intermediate set of q sent so far by p. The score-based improvement impact of q at peer p, denoted by $IScore(T_{cur}, T_{old})$ is computed as*

$$IScore(T_{cur}, T_{old}) = \frac{\sum_{d \in T_{cur}} q.f(d, q.\bar{q}) - \sum_{d' \in T_{old}} q.f(d', q.\bar{q})}{k} \quad (2)$$

Note that in Formula 2, we divide by k instead of $\|T_{cur} - T_{old}\|$ because we do not want that $IScore(T_{cur}, T_{old})$ be an average which would not be very sensitive to the values of scores. The score-based improvement impact values are in the interval $[0, 1]$.

In QUAT, the minimum value that must reach the improvement impact before a peer sends newly received intermediate results to its parent is initially set by the application and it is the same for all peers in the system. This threshold decreases as the query execution progresses. Using a dynamic threshold avoids the blocking problem of a static threshold when results having higher scores are bubbled up before those of lower score. Thus, we guarantee that low score results even tough they are in the final top-k results will not be returned at the end of the query execution.

To use a dynamic threshold approach, we need to compute the threshold value dynamically. We have identified two possible solutions for the dynamic threshold. The first one is to use an estimation of the query execution time. However, estimating the query execution time in large P2P system is very difficult because it depends on network dynamics, such as connectivity, density, medium access contention, etc., and the slowest queried peer. The second, more practical, solution is to use for each peer its local result set coverage to decrease the threshold. The local result set coverage of a peer for a given query is the proportion of peers in its sub-tree including itself which have already processed this query. We formalize this in Definition 3.

Definition 3 ***Peer's local result set coverage.*** *Given a top-k query, and $p \in \bar{P}$ (where $\bar{P}$ is the set of peers which received q), let $\mathcal{A}$ be the set of peers in the sub-tree whose root is p in the query q's forwarding tree. Let $\mathcal{E}$ be the set of peers in $\mathcal{A}$ which have already processed q locally. The local result set coverage of peer p for q, denoted by $Cov(\mathcal{E}, \mathcal{A})$, is computed using the following equation:*

$$Cov(\mathcal{E}, \mathcal{A}) = \frac{\|\mathcal{E}\|}{\|\mathcal{A}\|}$$

Peer's local result set coverage values are in the interval $[0, 1]$.

Computing the exact value of a peers local result set coverage incurs additional messages to the network, i.e. because each peer must send a message to its parent each time its local coverage result set value changes. To deal with this problem, we compute an estimation of this value instead of the exact value.

In our approach, the estimation is computed at the beginning by each peer based on the *ttl* received with the query and the average degree of peers in the system. This value is updated progressively as the peers in its sub-tree bubble up their results. Indeed, each peer includes in each response message sent to its parent the number of peers in its sub-tree (including itself) which have already processed the query locally and the total number of peers in its sub-tree including itself. This couple of values is used in turn by its parent to estimate its local result set coverage. To decrease the improvement impact threshold used by a peer as the local result set coverage increases, we use a linear function that allows peers to set their improvement impact threshold for a given local result set coverage. Now let us define formally the threshold function.

Definition 4 ***Threshold Function.*** *Given a top-k query q and $p \in \bar{P}$ (where $\bar{P}$ is the set of peers which received q), the improvement impact threshold used by p during q's execution, is a monotonically decreasing function H such that:*

$$H : \begin{vmatrix} [0,1] \rightarrow [0,1] \\ \\ x \quad \mapsto -\alpha * x + \alpha \end{vmatrix} \tag{3}$$

with $\alpha \in [0, 1[$. Notice that x is a peer's result set coverage at given time and α the initial improvement impact threshold (i.e. $H(0) = \alpha$).

5 Distributed Routing Indices

In this section, we first describe how to construct distributed routing indices and then how to maintain them.

5.1 Routing Indices Construction

A description routing index (or routing index for short) allows a peer to determine the priority of neighbors for sending a query when there is high query load in the system. It also help peers to avoid forwarding a query to some neighbors if their results for this query are not likely to bring anything to current top-k result set. Routing index is a data structure that, given a query, returns a list of neighbors, ranked according to their potential to answer the query. Let us now explain how these indices are created by peers. When a new peer p_i joins the system, it exchanges its own description with those of its direct neighbors and these neighbors' direct neighbors (i.e peers which are 2 hops from p_i). Using these descriptions, the peer p_i builds a description table of its neighborhood. This table contains the identifier of each neighbor p_j of p_i and the aggregation of local descriptions of p_j and p_j's direct neighbors. Descriptions tables are used as routing indices for top-k query processing.

5.2 Maintaining Routing Indices

Updates of data owned by a peer may cause the modification of its description. Therefore it is necessary that this modification be propagated to the neighbors to ensure accuracy of results returned to the user. A naive solution to maintain descriptions up-to-date is to broadcast an update message containing the new description of the peer and having $ttl = 1$ to all its direct neighbors. Each neighbor which receives this update message, decreases the ttl of this message and sends it in turn to its neighbors (except to a peer from which it receives this message) until the ttl value reaches 0. The maintenance of a routing index after a modification in the peer's description is done in $O(\varphi + \varphi^2)$ messages where φ is the average degree of peers in the system.

For efficiency reasons, we may choose not to send updates when the difference between the old and the new description of a peer is not significant. By not sending minor updates, we can trade update cost for accuracy of the index.

Finally, a special update occurs in the case of churn of peers. When a peer pi detects the disconnection of one of its neighbor p_j, p_i updates its routing index by removing the row for p_j. Then, it informs its direct neighbors by sending them an update message with $ttl = 1$. Each neighbor which receives this update message, decreases ttl by one and sends it in turn to its neighbors (until ttl reaches 0).

6 Performance Evaluation

In this section, we evaluate the performance of QUAT through simulation using PeerSim [10], an open source, Java based, P2P simulation framework. First, we

Table 1. Simulation parameters

Parameters	**Values**
Latency	Normally distributed random number, Mean=200 ms, Variance=100
Number of peers	10000 peers
Average degree	4
ttl	9
k	20
Query arrival rate	50 queries per seconds

describe our simulation setup, the metrics used for performance evaluation, the baseline top-k query processing approaches and the datasets used for experiments. Then, we study the effect of the query arrival rate on the performance of QUAT, and show how it scales up. Next, we investigate the effect of peers failures on the correctness of QUAT.

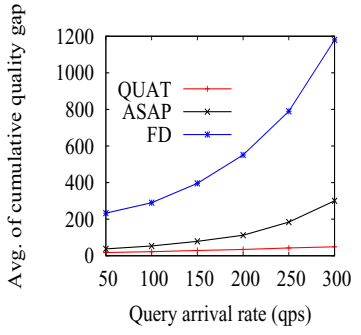
6.1 Setup

We implemented our simulation using the PeerSim simulator. We conducted our experiments on a machine with a 2.4 GHz Intel Pentium 4 processor and 2GB memory. The simulation parameters are shown in Table 1. We use parameters values which are typical of P2P systems [14]. The latency between any two peers is a normally distributed random number with a mean of 200 ms. Since users are usually interested in a small number of top results, we set $k = 20$ as default value. In our experiments we vary the network size from 1000 to 10000 peers . In order to simulate high heterogeneity, we set peers' capacities in our experiments, in accordance to [14] which measures peer capacities in the P2P system. Based on the results of [14], we generate around 10% of low-capable, 60% of medium-capable, and 30% of high-capable peers. The highly-capable peers are 3 times more capable than medium-capable peers and still 7 times more capable than low-capable ones. Each experiment is run for 2 hours, which are mapped to simulation time units. In all our experiments, we use $H(x) = -0.2x + 0.2$ as threshold function.

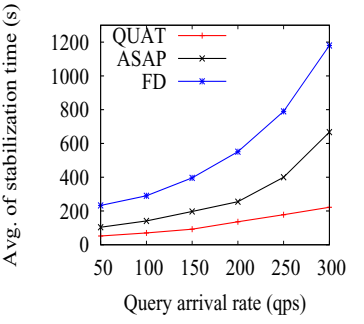
6.2 Dataset

We conduct our experiments using HTTP server logs dataset. The Internet Traffic Archive[3] provides a huge HTTP server log with about 1.3 billion HTTP requests from the 1998 FIFA soccer world championship. We aggregated the information from this log into a relational table with the schema $Log(interval, userid, bytes)$, aggregating the traffic (in bytes) for each user within one-day intervals. This dataset is horizontally partitioned evenly among peers of the P2P system. Queries ask for the top-k active users, i.e. the k users with the highest traffic at given interval (like "June 1").

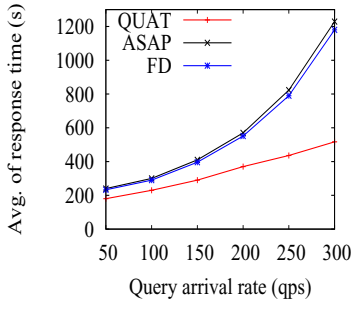
[3] http://ita.ee.lbl.gov



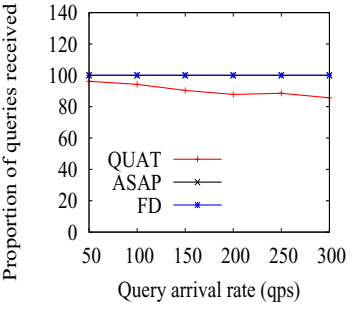
(a) Cumulative quality gap vs. Query rate



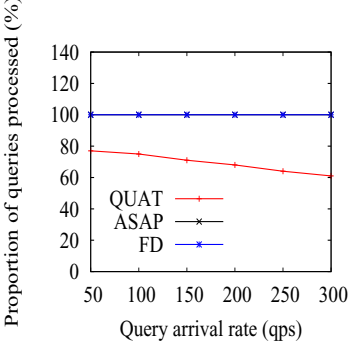
(b) Stabilization time vs. Query rate



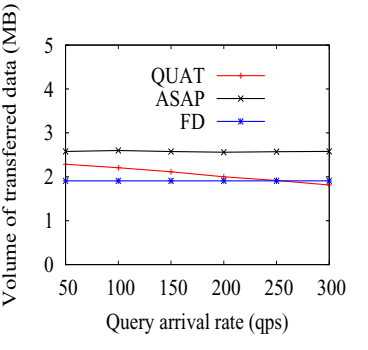
(c) Response time vs. Query rate



(d) % queries received vs. Query rate



(e) % queries processed vs. Query rate



(f) Volume of transferred data vs. Query rate

Fig. 2. Impact of query loads on QUAT performance

6.3 Metrics

In our experiments, to evaluate the performance of QUAT and that of baseline approaches, we use the following metrics:

(i) **Cumulative quality gap over a period:** see Section 3 for the definition.
(ii) **Stabilization time over a period:** see Section 3 for the definition.
(iii) **Response time over a period:** We report on the average response time of all queries submitted in the system over a given period. The response time is the time the query initiator has to wait until the top-k query execution is finished.
(iv) **Proportion of queries received per peer:** We report on the number of queries received in average by a peer over the number of queries submitted in the system over a given period.
(v) **Proportion of queries processed per peer:** We report on the number of queries executed locally in average by a peer over the number of queries received by a peer over a period.
(vi) **Communication cost:** We measure the communication cost in terms of number of answer messages and volume of data which must be transferred over the network in order to execute a top-k query.
(vii) **Accuracy of results:** We define the accuracy of results as follows. Given a top-k query q, let V be the set of the k top results owned by the peers that received q, V' be the set of top-k results which are returned to the user as the response of the query q. We denote the accuracy of results by ac_q and define it as:

$$ac_q = \frac{\|V \cap V'\|}{\|V\|}$$

6.4 Baseline Approaches

In unstructured P2P systems, Fully Distributed (FD) [1] and As Soon As Possible (ASAP) [6] are baseline approaches for top-k processing over horizontally partitioned data stored on peers. In FD, each peer that receives the query executes it locally (i.e. selects the k top scores), and waits for its children's results. After receiving all its children score-lists, the peer merges its k local top data items with those received from its children and selects the k top scores and sends the result to its parent. Unlike FD, in ASAP, a peer does not wait for all its children results before bubbling up results to its parent. Each peer (except the query originator) returns to its parent its intermediate results that have better qualities and thus may be in the final top-k.

6.5 Performance Results

In this section we present the results of our experimentation. Due to space limitations, we only present the main results.

Effect of arrival query rate. We study the effect of the query arrival rate on the performance of QUAT. For this, we ran experiments using the HTTP logs dataset to study cumulative quality gap, stabilization time, response time, proportion of queries received, proportion of queries processed and volume of

transferred data while increasing the query arrival rate in the system from 50 to 300. Note that the other simulation parameters are set as in Table 1.

Figures 2(a) and 2(b) show respectively how cumulative quality gap and stabilization time over a period of 2 hours increase with the query arrival rate. The results show that the cumulative quality gap of QUAT is always much smaller than that of ASAP and FD, which means that QUAT returns much faster high quality results than ASAP and FD. The results also show that the stabilization time of QUAT is always much smaller than that of ASAP and FD. The reason is that in QUAT, peers prioritize the execution of queries that can produce high quality results. Figure 2(c) show that the response time of QUAT over a period of 2 hours is always much better than that of ASAP and FD. The main reason is that in QUAT, peers do not execute incoming queries for which they do not have interesting data, which helps peers to save their resources.

Figures 2(d) and 2(e) show that the proportion of queries received and the proportion of queries processed by peers over 2 hours decrease while increasing the query arrival rate. The reason is that in QUAT, as the query rate increases, peers reduce the maximum number of connections that they can open simultaneously for a query. In addition, they use their knowledge of the descriptions of their neighbors to avoid sending queries to some neighbors. Moreover they exploit their local description to avoid executing locally some queries.

Figure 2(f) shows the volume of the increase of transferred data vs. query arrival rate. The results show that the volume of transferred data of QUAT is always higher than that of ASAP. The results also show that the difference between QUAT and FD's volume of transferred is not significant.

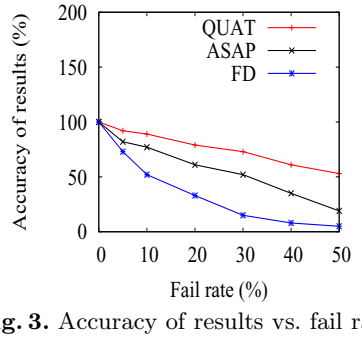


Fig. 3. Accuracy of results vs. fail rate

Effect of peers failures. In this section, we investigate the effect of peers failures on the accuracy of top-k results. In our tests, we vary the value of fail rate and investigate its effect on the accuracy of top-k results. Figure 3 shows the accuracy for QUAT, ASAP and FD while increasing the fail rate, with the other parameters set as in Table 1. Peers' failures have less impact on QUAT than ASAP and FD. The reason is that QUAT returns high-score results to the user very quickly. However, when increasing the fail rate in ASAP and FD, the accuracy of top-k results decreases significantly because some score-lists are lost.

Indeed, in FD, each peer waits for results of its children so in the case of a peer failure, all the score-lists received so far by that peer are lost.

7 Related Work

Efficient processing of top-k queries is both an important and hard problem that is still receiving much attention [17,13]. Several works have dealt with top-k query processing in centralized database management systems [15,9]. In distributed systems [4], previous work on top-k processing has focused on vertically distributed data over multiple sources, where each source provides a ranking over some attributes. The majority of the proposed approaches try to improve some limitations of the Threshold Algorithm (TA) [7]. Following the same concept, there exist some previous work for top-k queries in P2P over vertically distributed data. In [3], the authors propose algorithm called "Three-Phase Uniform Threshold" (TPUT) which aims at reducing communication cost by pruning away intelligible data items and restricting the number of round-trip messages between the query originator and other nodes. Later, TPUT was improved by KLEE [11]. KLEE uses the concept of bloom filters to reduce the data communicated over the network upon processing top-k queries. It brings significant performance benefits with small penalties in result precision. However, theses approaches assume that data is vertically distributed over the nodes whereas we deal with horizontal data distribution.

For horizontally distributed data, there has been little work on P2P top-k processing. In [1], the authors present FD, a fully distributed approach for top-k query processing in unstructured P2P systems. Recently, FD was improved by ASAP [6]. We have briefly introduced FD and ASAP in section 6.4.

In [2], the authors present an index routing based a top-k processing technique for super-peer networks organized in an HyperCuP topology which tries to minimize the number of transfer data. The authors use queries statistics to maintain the indices built on super-peers. However, the performance of this technique depends on the query distribution.

Zhao *et al.* [19] use a result caching technique to prune network paths and answer queries without contacting all peers. The performance of this technique depends on the query distribution. They assume acyclic networks, which is restrictive for unstructured P2P systems.

There have been many works to deal with the problem of query load balancing by trying to distribute the load fairly over the peers of the system, e.g. [5]. However, in the current paper, our objective is not to balance the load, but to take it into account for reducing the user waiting time.

8 Conclusion

In this paper, we addressed the problem of top-k query processing in overloaded P2P systems. The objective is to reduce the user waiting time by returning high quality intermediate results as soon as possible, while avoiding high network

traffic. For this, we revisited the problem of top-k query processing by considering two new metrics to complement response time: stabilization time and cumulative quality gap. Then, we proposed QUAT, an efficient algorithm that dynamically adapts to peer query loads in order to return to the user top-k results as soon as possible. QUAT allows users to progressively see the evolution of their query execution by receiving high quality intermediate results. We validated QUAT through extensive experimentation. The results show that QUAT significantly outperforms baseline algorithms by providing quickly high quality to users and by returning final top-k result to users in much better times. Finally, the results demonstrate that in the presence of peers' failures unlike baseline algorithms, QUAT provides top-k results with good accuracy.

References

1. Akbarinia, R., Pacitti, E., Valduriez, P.: Reducing network traffic in unstructured p2p systems using top-k queries. Distributed and Parallel Databases 19(2-3), 67–86 (2006)
2. Balke, W.-T., Nejdl, W., Siberski, W., Thaden, U.: Progressive distributed top k retrieval in peer-to-peer networks. In: Proceedings of Int. Conf. on Data Engineering (ICDE), pp. 174–185 (2005)
3. Cao, P., Wan, Z.: Efficient top-k query calculation in distributed networks. In: Proceedings of Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 206–215 (2004)
4. Chaudhuri, S., Gravano, L., Marian, A.: Optimizing top-k selection queries over multimedia repositories. IEEE Transactions on Knowledge Data Engineering (TKDE) 16(8), 992–1009 (2004)
5. Datta, A.: Load balancing in peer-to-peer overlay networks. In: Encyclopedia of Database Systems, pp. 1627–1632. Springer, US (2009)
6. Dedzoe, W.K., Lamarre, P., Akbarinia, R., Valduriez, P.: Asap top-k query processing in unstructured p2p systems. In: Proceedings of IEEE Int. Conf on Peer-to-Peer Computing (P2P), pp. 187–196 (2010)
7. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Proceedings of Symposium on Principles of Database Systems (PODS), pp. 102–113 (2001)
8. Hayek, R., Raschia, G., Valduriez, P., Mouaddib, N.: Summary management in p2p systems. In: Proceedings of Int. Conf on Extending Database Technology (EDBT), pp. 16–25 (2008)
9. Hristidis, V., Koudas, N., Papakonstantinou, Y.: Prefer: a system for the efficient execution of multi-parametric ranked queries. In: Proceedings of ACM. Int Conf. on Management of Data (SIGMOD), pp. 259–270 (2001)
10. Jelasity, M., Montresor, A., Jesi, G.P., Voulgaris, S.: The Peersim simulator, http://peersim.sf.net
11. Michel, S., Triantafillou, P., Weikum, G.: Klee: A framework for distributed top-k query algorithms. In: Proceedings of Int. Conf. on Very Large Data Bases (VLDB), pp. 637–648 (2005)
12. Ooi, B.C., Shu, Y., Tan, K.-L.: Relational data sharing in peer-based data management systems. SIGMOD Record 32(3), 59–64 (2003)

13. Radwan, A., Popa, L., Stanoi, I.R., Younis, A.A.: Top-k generation of integrated schemas based on directed and weighted correspondences. In: Proceedings of ACM Int. Conf. on Management of data (SIGMOD), pp. 641–654 (2009)
14. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of the Multimedia Computing and Networking Conference (MMCN 2002) (2002)
15. Shmueli-Scheuer, M., Li, C., Mass, Y., Roitman, H., Schenkel, R., Weikum, G.: Best-effort top-k query processing under budgetary constraints. In: Proceedings of Int. Conf. on Data Engineering (ICDE), pp. 928–939 (2009)
16. Ventresque, A., Lamarre, P., Cazalens, S., Valduriez, P.: Representation optimiste de contenus dans les systems p2p. In: Actes de la conference Bases de Donnes Avances (BDA), pp. 1–20 (2009)
17. Vlachou, A., Doulkeridis, C., Kotidis, Y., Nørvåg, K.: Reverse top-k queries. In: Proceedings of Int. Conf. on Data Engineering (ICDE), pp. 365–376 (2010)
18. Vlachou, A., Doulkeridis, C., Nørvåg, K., Vazirgiannis, M.: On efficient top-k query processing in highly distributed environments. In: Proceedings of ACM. Int. Conf. on Management of Data (SIGMOD), pp. 753–764 (2008)
19. Zhao, K., Tao, Y., Zhou, S.: Efficient top-k processing in large-scaled distributed environments. Data and Knowledge Engineering 63(2), 315–335 (2007)

Top-k Query Evaluation in Sensor Networks with the Guaranteed Accuracy of Query Results

Baichen Chen[1], Weifa Liang[1], and Geyong Min[2]

[1] Research School of Computer Science
The Australian National University
Canberra, ACT 0200, Australia
[2] School of Computing
University of Bradford
Bradford, BD7 1DP, United Kingdom

Abstract. In many applications of sensor networks including environmental monitoring and surveillance, a large volume of sensed data generated by sensors needs to be either collected at the base station or aggregated within the network to respond to user queries. However, due to the unreliable wireless communication, robust query processing in such networks becomes a great challenge in the design of query evaluation algorithms for some mission-critical tasks. In this paper we propose an adaptive, localized algorithm for robust top-k query processing in sensor networks, which trades off between the energy consumption and the accuracy of query results. In the proposed algorithm, whether a sensor is to forward the collected data to the base station is determined in accordance with the calculation of a proposed local function, which is the estimation of the probability of transmitting the data successfully. We also conduct extensive experiments by simulations on real datasets to evaluate the performance of the proposed algorithm. The experimental results demonstrate that the proposed algorithm is energy-efficient while achieving the specified accuracy of the query results.

1 Introduction

In recent years, technological advances have made it become possible to deploy large-scale sensor networks, consisting of hundreds or thousands of inexpensive sensors in an ad-hoc fashion, for environmental monitoring and security surveillance purposes [1,16]. In these applications, a large volume of sensing data generated by sensors needs to be either collected at the base station or aggregated within the network to respond to user queries. The sensor network thus can be treated as a *virtual database* by the database community [14]. The processing of queries in wireless sensor network includes the skyline query [4], top-k query [22,23,5,6,11], join query [9,20,21], and so on. Top-k query is a fundamental operator in databases that searches for very important objects according to the object rankings obtained by a variety of ranking techniques. Efficient processing of top-k query is crucial in many information systems that comprise a large amount of data [8]. Top-k query in a sensor network is to return the k

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 156–171, 2011.

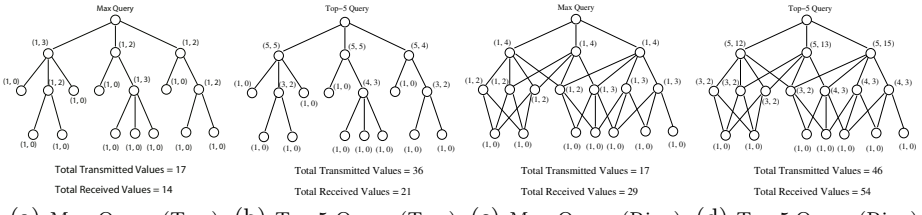
points with largest values to the base station, where a point is referred to as the sensed value and the ID of its generator (sensor). Wireless sensor networks that support top-k queries can be used to not only monitor the data generated by sensors in no time but also perform further data analysis for decision making. One such an application scenario is that the ornithologists who study the behaviors of various bird species in a given region forest are interested to know where the birds are most likely to gather [22]. To do so, they place the bird feeders at different locations in the monitored region and install one sensor at each feeder to count the number of birds at that feeder periodically. The result of the top-k query can assist the ornithologists to determine where the birds are likely to be attracted. For example, a top-k query can inquire which feeders attract the maximum number of birds. Thus, the ornithologists can observe the bird behaviors at a few places where the most attractive feeders are located.

A paramount concern in processing queries in energy-constrained wireless sensor networks is the energy conservation in order to prolong the network lifetime, because it usually is impractical to recharge the batteries that power the sensors. In addition, a query result with a certain degree of accuracy is acceptable, while the query results are typically computed by in-network processing. The existing in-network processing algorithms are mainly based on the tree routing structure, which include the ones in [13,15,24,25] for aggregate query, and the ones in [23,6,10] for top-k query, etc. However, the failure rates of wireless communication in wireless sensor networks are relatively high (up to 30% loss rate in common [27]), and each lost message at a sensor causes the loss of all the collected data from the subtree rooted at the sensor. As a result, it is not uncommon that 85% of sensed values are lost in a multi-hop sensor networks, causing significantly answer inaccuracy and compromising the monitoring quality [18]. To overcome the shortcoming of the tree structure in the accuracy of query results, several algorithms including algorithm `FATE-CSQ` in [12] have been proposed, which make use of the feedback-retransmission mechanism, i.e., if a transmission is failure, the parent sends the feedback messages to the children and the children retransmit their messages again. However, such algorithms result in high message complexity and long delay in message delivery. Gobriel et al [7] proposed an algorithm `RideSharing`, in which each sensor maintains two types of parents: the primary and several backup parents. If the primary parent does not receive the message from a child, it would send a vector to the other backup parents, asking for them to forward the message. To ensure that all the parents of a sensor can overhear the vector, it is required that all the parents and this sensor form a *clique*, i.e., each of them is located within the transmission range of each other. Although algorithm `RideSharing` avoids multiple retransmissions of the messages, it suffers high message complexity and long delay too, and furthermore, the clique may not exist in real networks.

Besides the mentioned tree-based algorithms, several researchers have proposed multi path-based and hybrid-based algorithms to deal with the aggregation queries in wireless sensor networks with high failure rates [3,2,18,17,26]. Considine et al [3,2] and Nath et al [18] proposed the multi path-based

methods, based on the multi-ring routing structure for aggregation queries, in which the sensors are partitioned into different levels according to the number of hops between them and the base station. Therefore, the data transmission is performed level by level towards the base station. In the transmission by using the multi-ring structure, each sensor sends its messages to all of its neighbors at the level closer to the base station, rather than the single parent in the tree routing structure. In this paper, this approach is referred to as algorithm `SD` (`Synopsis Diffusion`) from [18]. The multi-ring structure is efficient in terms of energy consumption for some aggregation queries such as *MAX, MIN* and *SUM*, because each sensor aggregates the received data and broadcasts the aggregate result of the same size as each received data to its neighbors. Therefore, the transmission energy consumption on the multi-ring structure is almost the same as that on the tree structure. However, in dealing with complicated queries like top-k query and skyline query, a number of points rather than a partial result need to be sent to the base station as part of the query result, which means that the duplication of points will significantly increase the transmission and reception energy consumptions. This leads to that the sensors run out of their energy quickly, thereby reducing the lifetime of the sensor network. We here use an example to illustrate this. Fig. 1(a)-(d) show the number of sent and received points by the sensors when the tree based approach and algorithm `SD` are applied to answering a *MAX* query and a top-5 query, respectively. Each sensor has a labeled tuple, in which the first component is the number of points sent by the sensor, and the second one is the number of points received at the sensor. Initially, each sensor contains one point. To answer the *MAX* query, each sensor broadcasts a point with largest value among the received and its own points to its neighbors, while each sensor broadcasts 5 points with largest values among the received and its own points to its neighbors to answer the top-5 query. It can be seen that for the *MAX* query, algorithm `SD` has the same number of sent points and a few more received points, while for top-5 query, the number of sent and received points of algorithm `SD` is much more than those on tree topology.

To utilize the advantage brought by the tree and ring routing structures, Manjhi et al [17] proposed an algorithm `TD (Tributary-Delta)` for aggregation query processing, which is a hybrid approach, i.e., it adopts the routing structure for data aggregation to achieve the optimal performance. Algorithm `TD` tries to



(a) Max Query (Tree) (b) Top-5 Query (Tree) (c) Max Query (Ring) (d) Top-5 Query (Ring)

Fig. 1. Answering a Top-k Query and a Max Query on Tree and Ring Topologies

overcome the problems of tree and ring topologies by combining the best features of both topologies and tradeoffing the energy consumption and robustness of query processing. The core idea of algorithm `TD` is to divide the sensors into two categories. The sensors far away from the base station are called T sensors, which send their data to a single parent, while the sensors near to the base station are called M sensors, which send their data to multiple neighbors closer to the base station. The network thus is organized in regions to implement one of the two structures respectively. The region consists of the T sensors is called *Tributary* region, while the region consists of M sensors and the base station is called *Delta* region. The base station maintains the percentage of the sensors that contribute their data to the final result and decides whether to shrink or expand the *Delta* region for the future queries. If the percentage is below the user given threshold, the *Delta* region is expanded to improve the robustness; otherwise, the *Delta* region is shrunk to save energy. The adaptation of the routing structure is executed by changing the label of a sensor (T or M). However, algorithm `TD` seems not to be applicable to top-k query due to the following concerns. Firstly, the points are only sent to a parent at the T sensors, therefore the accuracy of query results will become low if the link failure rate is high. Secondly, the re-classification of sensors is purely derived from the statistics of the failure links in the transmission of current query evaluation. If the data distribution and the unpredictable status of the wireless links in the next period are different from the previous ones, such a re-classification may not improve the accuracy of the results at all. Thirdly, the sensors near to the base station usually consume more energy than the sensors far away from the base station. In algorithm `TD`, the M sensors send and receive as many points as the sensors in algorithm `SD`, which makes the M sensors run out of their energy quickly and disconnection between the base station and other sensors. Thus the sensor network is no more functioning. Lastly, the maintenance of the network structure may be expensive in energy consumption by broadcasting the messages for adapting the structure.

In this paper we deal with top-k query evaluation in wireless sensor networks efficiently and effectively. Our main contributions are as follows. We first analyze the drawbacks of applying existing algorithms for answering top-k queries, followed by giving a new definition of the accuracy of top-k query results. We then propose a localized, energy-efficient query evaluation algorithm tradeoffing between the energy consumption and the accuracy of query results. We finally conduct extensive experiments by simulations on real sensing datasets to evaluate the performance of the proposed algorithm. The experimental results show that the proposed algorithm outperforms existing ones in terms of energy consumption under the given constraint of the accuracy of results.

2 Preliminaries

2.1 System Model

We consider a sensor network consisting of n stationary sensors, randomly deployed in a region of interest, each measuring a numerical value. For each

sensor v, $v.id$ is the ID of sensor v. Each *point* p is represented by a tuple $< p.sid, p.val >$, where $p.sid$ is the ID of sensor generating point p, and $p.val$ is the sensing (reading) value. Assume that $p.sid$ as well as $p.val$ is represented by 4 bytes. Thus, a point p is represented by 8 bytes in total. There is a base station with unlimited energy supply, which serves as the gateway between the sensor network and users. Each sensor can communicate with the other sensors within its transmission range, where the transmission range of all sensors in the network is identical. Denote by f the probability of a link failures during the wireless radio communication, i.e., the transmission through a link is failed with probability f, $0 \leq f \leq 1$. To transmit a message containing l bytes from one sensor to its neighbors, the amount of transmission energy consumed at the sender is $\rho_t + l * R$, while the amount of reception energy consumed at the receiver is $\rho_r + l * r$, where ρ_t and ρ_r are the sum of energy overhead on handshaking and sending and receiving the header of the message, and R and r are the amounts of transmission and reception energy per byte. We assume that the computation energy consumption on sensors can be ignored, because in practice it is several orders of magnitude less than the communication energy consumption, e.g., the authors in [14,19] claimed that the transmission of a bit of data consumes as much energy as executing 1,000 CPU instructions.

2.2 Query Structure

Denote by $N(v)$ the set of *neighbors* of sensor v. Each sensor can transmit points to the base station through one or multi-hop relays. We define the *distance* of v to the base station as the *minimum number of hop relays* from sensor v to the base station. The sensors in the network can be divided into several *levels*, and the sensors at the same level are also indexed. According to their distances to the base station, $v_{i,j}$ is referred to as the sensor with jth smallest ID at the ith level and V_i is the set of sensors at the ith level of the network. The base station is in V_1 (the base station is $v_{1,1}$), and the number of hops from a sensor $v_{i,j}$ to the base station is $i-1$. Let $Up(v_{i,j})$ be the set of *upstream sensors* of $v_{i,j}$, which is the set of neighbors of v in V_{i-1}, i.e., $Up(v_{i,j}) = \{v \mid v \in V_{i-1}, v \in N(v_{i,j})\}$. Similarly, denote by $Down(v_{i,j}) = \{v \mid v \in V_{i+1}, v \in N(v_{i,j})\}$ the set of *downstream sensors* of $v_{i,j}$. A sensor v is defined as a *partner* of sensor $v_{i,j}$ from sensor u, such that v and $v_{i,j}$ have the same downstream sensor u, i.e., the set of *partner* of sensor $v_{i,j}$ from sensor u is $Par(v_{i,j})_u = \{v \mid u \in Down(v_{i,j}), u \in Down(v)\}$. In other words, a sensor in $Up(u)$ is a partner of any sensor in $Up(u)$ from sensor u. $v_{i,j}$ is also a partner of itself. Thus $Par(v_{i,j}) = \bigcup_{u \in Down(v_{i,j})} Par(v_{i,j})_u$ is the set of partners of sensor $v_{i,j}$ from all its downstream sensors. A sensor v is defined as the *ancestors* of sensor $v_{i,j}$ if the points generated at sensor $v_{i,j}$ can be sent to sensor v through one or multi-hop relays, while a sensor v is defined as the *descendants* of $v_{i,j}$ if the points from v can be sent to $v_{i,j}$ through one or multi-hop relays. The sets of ancestors and descendants of sensor $v_{i,j}$ are referred to as $Anc(v_{i,j})$ and $Des(v_{i,j})$, respectively. In addition, denote by $Down(v_{i,j})_u$ the subset of downstream sensors of sensor $v_{i,j}$, in which the sensors are the ancestors of sensor u, i.e., $Down(v_{i,j})_u = \{v \mid v \in Down(v_{i,j}), v \in Anc(u)\}$. In

the previous algorithms using the multi-ring structure, the points are transmitted from a sensor to all its upstream sensors and are regarded as the duplications of the points. The points thus are transmitted to the base station through multi-paths, which significantly increases the accuracy of the query results.

2.3 Accuracy of Top-k Query Results

The accuracy of the query results in previous studies is measured by the ratio of the obtained results to the real results. However, such a measurement may not be suitable for top-k queries. The reason is as follows. Assume that the base station issues a top-100 query to a sensor network of 1,000 sensors and receives the top-100 result from its 700 sensors, and the points from the rest of the 300 sensors are lost within the transmission due to link failures. However, due to the skew data distribution, the obtained top-100 result from the 700 sensors is the actual top-100 result from the 1000 sensors. If the data distribution is changed, e.g., the actual top-100 points all come from the 300 sensors, the obtained result and the actual result are completely different. In this case, the comparison between the obtained results and the actual results does not reflect the actual number of the points lost during the data transmission. To this end, we thus propose another metric to measure the accuracy of top-k query results that will not be affected by the data distribution as follows.

Denote by P the set of points in the sensor network. If there are x points in P which are actually used to determine the results, $\frac{x}{|P|}$ is defined as the accuracy of the top-k results, where the x points consist of two types of points: the points transmitted to the base station successfully, and the points failed to be transmitted to the base station because they are discarded by the sensors receiving at least k points with higher values than theirs. For each query, assume that an expected *threshold ratio* θ is given in advance, and at least $\theta * |P|$ points can be used for determining the results. The *single point threshold* θ_p is the probability of a point that is one of the x points. Denote by Ev_x the event that x points are used to determine the query result. Therefore, the accuracy of top-k result is

$$\begin{aligned}\theta &= Pr(Ev_{|P|}) + Pr(Ev_{|P|-1}) + \ldots + Pr(Ev_{\lceil |P|\theta \rceil}) \\ &= \theta_p^{|P|} + \theta_p^{|P|-1}(1-\theta_p)|P| + \ldots + \binom{|P|}{\lceil |P|\theta \rceil}(\theta_p^{|P|-\lceil |P|\theta \rceil})(1-\theta_p)^{\lceil |P|\theta \rceil} \\ &= \sum_{i=0}^{|P|-\lceil |P|\theta \rceil+1} \binom{|P|}{i}\theta_p^{|P|-i}(1-\theta_p)^i. \end{aligned} \quad (1)$$

The value of θ_p in Eq. (1) can be estimated provided that θ is given, and if each point used for the query result has a probability no less than θ_p, then the accuracy of the query result will be no less than θ. Note that due to the uncertainlity of the link failures, it cannot guarantee whether the accuracy of each top-k query result meets the specified threshold.

We now give the problem statement. Given a wireless sensor network $G(V, E)$, V is the set of sensors and the base station, and E is the set of links. Assume

that each sensor contains a point initially. Let P be the set of points generated by all sensors. Assume that the average link failure rate is f, which means that a sensor v in the WSN transmits data to its neighbors, the neighbors cannot receive the correct data with probability $f(v)$ on average, and the query result accuracy threshold θ is given, too. The *robust top-k query evaluation* is to return the k points with the largest values with the query result accuracy being no smaller than θ. If the link failure rate is too high to meet the accuracy threshold θ, the top-k query evaluation will return the results as accurate as possible.

3 Robust Top-*k* Query Evaluation Algorithm

In this section we propose a novel localized evaluation algorithm for robust top-k query to tradeoff the energy consumption and the query result accuracy. The proposed algorithm is as follows. Built upon the query structure, a sensor $v_{i,j}$ first collects the local information (i.e., its neighboring sensors). To respond to a top-k query, each sensor $v_{i,j}$ performs different operations according to whether it has downstream sensors. If there is no downstream sensors, $v_{i,j}$ just broadcasts its points to its upstream sensors; otherwise, it receives the points from all its downstream sensors and puts the points with the same values and the generated sensors into the same group. Sensor $v_{i,j}$ examines the groups one by one to determine whether to broadcast the points of groups to its upstream sensors. Having examined all the groups, $v_{i,j}$ broadcasts some of the received points to its upstream sensors. To determine whether to broadcast a received point, $v_{i,j}$ calculates a value of a local function. If the value of the function for the received point is less than the threshold θ_p, the point is forwarded to its upstream sensors; otherwise, the point is discarded. The calculation of the local function is based on the local information of sensor $v_{i,j}$ and its information so far. In the following, we first describe how each sensor collects the local information and then propose the algorithm `Robust Top-`k `Query Evaluation` (`RTE` in short). Note that the proposed algorithm is a localized algorithm, which is preferable by distributive sensor networks.

3.1 Local Information Collection

We show how a sensor $v_{i,j}$ collects the local information of its neighboring sensors. Firstly, each sensor $v_{i,j}$ broadcasts its ID $v_{i,j}.id$ and the number of its upstream sensors $|Up(v_{i,j})|$ to its downstream sensors, and at the same time it also received the same information from its upstream sensors. In other words, each sensor $v_{i,j}$ broadcasts $\{v_{i,j}.id, |Up(v_{i,j})|\}$ to its downstream sensors and receives $\bigcup_{v \in Up(v_{i,j})}\{v.id, |Up(v)|\}$ from its upstream sensors. Secondly, sensor $v_{i,j}$ broadcasts $\bigcup_{v \in Up(v_{i,j})}\{v.id, |Up(v)|\}$ to its upstream sensors. It also receives the information broadcast from all its downstream sensors. Having received the information from one of downstream sensor u , i.e., $\bigcup_{v \in Up(u)}\{v.id, |Up(v)|\}$, $v_{i,j}$ obtained the information of partners in $Par(v_{i,j})_u$. Therefore, sensor $v_{i,j}$ obtains the IDs and the number of upstream sensors of all its partners, i.e., $\bigcup_{v \in Par(v_{i,j})}\{v.id, |Up(v)|\}$. Finally, $v_{i,j}$ makes use of the information to calculate the local function and determines whether to forward the received points. Note that the local information collection is only executed once.

3.2 Query Evaluation Algorithm

We propose the evaluation algorithm RTE for robust top-k query evaluation as follows. For each sensor $v_{i,j}$, the set of its upstream sensors is partitioned into two disjoint subsets: The first $\lceil |Up(v_{i,j})| * f \rceil$ sensors with smaller IDs are *appointed* to forward the points from $v_{i,j}$, while the other sensors will determine whether to forward the points by calculating a *local function*. Note that each sensor can easily know whether it is appointed by one of its downstream sensors because it stores the IDs of all its partners. The detailed procedure is as follows.

If sensor $v_{i,j}$ does not have any downstream sensors, it broadcasts the points to all its upstream sensors; otherwise, it checks the received points one by one. There are different groups of points at $v_{i,j}$, containing the same points from different downstream sensors. The points in the group $Gp(p)_{v_{i,j}}$ are the points whose values are $p.val$ and their generated sensors are $p.sid$. $v_{i,j}$ contains k different groups of points $Gp(p_1)_{v_{i,j}}, \ldots, (p_k)_{v_{i,j}}$, in which $p_1, \ldots, p_k$ are the k points with highest values among the points at sensor $v_{i,j}$, where for every two groups $Gp(p_x)_{v_{i,j}}$ and $Gp(p_y)_{v_{i,j}}$, $p_x.sid \neq p_y.sid$ or $p_x.val \neq p_y.val$. If sensor $v_{i,j}$ is appointed by any downstream sensors sending p_x to $v_{i,j}$ and p_x is in one of the top-k group, point p_x is marked as the point to be forwarded by $v_{i,j}$; otherwise, sensor $v_{i,j}$ calculates the function to determine whether to forward point p_x. The calculation of the local function will be introduced in the next section. If the value of the local function is not smaller than θ_p, point p_x is discarded by sensor $v_{i,j}$; otherwise, point p_x is marked to be forwarded. Having checked all the groups, sensor $v_{i,j}$ broadcasts the set of all marked points to its upstream sensors. The pseudo-code of algorithm RTE at each sensor is as follows.

Algorithm 1. Algorithm RTE(v)

```
begin
    collect the local information; receive θ and calculates θ_p;
    if v_{i,j} has no downstream sensors then
        broadcast the points to all the upstream sensors;
    else
        receive the points from the downstream sensors;
        group the points and obtains top-k points groups,
        G(p_1)_{v_{i,j}}, ..., G(p_k)_{v_{i,j}};
        foreach point p_x of group G(p_x)_{v_{i,j}} do
            if v_{i,j} is appointed by any downstream sensors sending p_x then
                p_x is marked to be forwarded;
            else
                calculate the local function;
                if the value of local function is smaller than θ_p then
                    p_x is marked to be forwarded;
        end
        broadcast the points that are marked to be forwarded;
end
```

3.3 Determination of Non-appointed Sensors

We then describe how a sensor $v_{i,j}$ determines whether forwarding a received point p if it is not appointed by any sender of p. The intuition of determination is that sensor $v_{i,j}$ calculates the probability of transmitting point p by its partners appointed to forward point p. If the probability is no less than the threshold, there is no need for it to forward point p; otherwise, sensor $v_{i,j}$ forwards point p. However, it is a bit difficult for sensor $v_{i,j}$ to calculate the probability of sending point p by the appointed partners, since $v_{i,j}$ only has the local rather than the global information. Therefore, an approximate value that is always smaller than that probability is designed for sensor $v_{i,j}$ as follows.

Denote by $pr(v)_p$ or $pr(V_{i-1})_p$ the probability of point p sent to sensor v or any sensor at the $i-1$th level, while $lpr(v)_p$ and $lpr(V_{i-1})_p$ are the local functions of sensor $v_{i,j}$ estimating the probability of that point p is sent to sensor v or any sensor at the $i-1$th level, respectively. Assume that point p is generated at sensor $v_{i',j'}$ and $v_{i,j}$ is the ancestor of $v_{i',j'}$. Recall that $Down(v_{i,j})_{v_{i',j'}}$ is the set of downstream sensors of $v_{i,j}$ and the ancestors of $v_{i',j'}$. In other words, point p can be transmitted from $v_{i',j'}$ to $v_{i,j}$ through sensor $u \in Down(v_{i,j})_{v_{i',j'}}$. Suppose that sensor $v_{i,j}$ receives point p successfully from m sensors $\{u_1, u_2, \ldots, u_m\} \subseteq Down(v_{i,j})_{v_{i',j'}}$. Sensor v is a partner of sensor $v_{i,j}$ that is also ancestor of sensor $v_{i',j'}$. Denote by $d(v)$ the number of downstream sensors of sensor v in $\{u_1, u_2, \ldots, u_m\}$. For a sensor v in $\bigcup_{t=1}^{m} Par(v_{i,j})_{u_t}$, sensor $v_{i,j}$ estimates the probability that sensor v receives point p is

$$lpr(v)_p = 1 - (1 - (1 - f)^{i'-i-1}(1 - f))^{d(v)}. \tag{2}$$

where f is the average of link failure rate and the value of $d(v)$ can be calculated by sensor $v_{i,j}$ as follows. Initially $d(v) = 0$. Having received point p from sensors $u_1, \ldots, u_m$, sensor $v_{i,j}$ increments $d(v)$ by 1 if v is in $Up(u_t)$, $1 \le t \le m$, and i' can be obtained from $p.sid$. Thus the value of $lpr(v)_p$ can be calculated locally. Assume that there are m' partners of $v_{i,j}$, $v_1, \ldots, v_{m'}$ in $\bigcup_{t=1}^{m} Par(v_{i,j})_{u_t}$ appointed to forward point p. Sensor $v_{i,j}$ calculates the value of $lpr(V_{i-1})_p$ by calculating the probability that all sensors $v_1, v_2, \ldots, v_{m'}$ send point p to their upstream sensors. From Eq. (2), we have

$$\begin{aligned} lpr(V_{i-1})_p &= 1 - \prod_{t=1}^{m'} (1 - lpr(v_t)_p(1 - f^{|Up(v_t)|})) \\ &= 1 - \prod_{t=1}^{m'} (1 - (1 - (1 - (1 - f)^{i'-i})^{d(v_t)})(1 - f^{|Up(v_t)|})), \end{aligned} \tag{3}$$

The value of $|Up(v_t)|$ can be obtained by sensor $v_{i,j}$ through local information collection and all the variables in Eq. (3) can be obtained by sensor $v_{i,j}$, thus the value of $lpr(V_{i-1})_p$ can be calculated locally. For each received point p, if sensor $v_{i,j}$ is not appointed, it calculates the value of $lpr(V_{i-1})_p$. If $lpr(V_{i-1})_p > \theta_p$, sensor $v_{i,j}$ discards point p; otherwise, sensor $v_{i,j}$ sends point p to all its upstream sensors even if it is not appointed to forward point p. When $lpr(V_{i-1})_p$ is

small, all the sensors need to forward the points to their upstream sensors to improve the probability no matter whether they are appointed. Consequently, the proposed algorithm is the same as algorithm SD. If the estimated probability is high, the non-appointed sensors discard the points directly. $lpr(V_{i-1})_p$ in Eq. (3) is determined by f, $|Up(v_t)|$, $d(v_t)$, and $i - i'$, while the values of f and $|Up(v_t)|$ reflect the physical condition of the sensor networks, e.g., the average link failure rate and the deployment of the sensors (the number of upstream sensors of a sensor). If f is large and $|Up(v_t)|$ is small, this means the chance to transmit point p successfully is unlikely, the value of $lpr(V_{i-1})_p$ will become small and consequently more sensors help is needed in order to forward point p to their upstream sensors. $d(v_t)$ is determined by the number of copies of point p received by sensor $v_{i,j}$ and the number of downstream sensors of sensor v_t. A smaller $d(v_t)$ implies that the number of copies of point p is small and loss of these copies will lead to the failure of the transmission of point p. When $d(v_t)$ is small, $lpr(V_{i-1})_p$ will be small as well and more sensors forward point p to increase the number of copies of p. $i - i'$ shows the number of hops from the generated sensor of point p ($v_{i',j'}$) to sensor $v_{i,j}$. The larger the $i - i'$ is, the more possible point p is in the top-k result because it has larger values than more points from the subtree rooted at sensor $v_{i,j}$. If $i - i'$ is large, $lpr(V_{i-1})_p$ will be small and the non-appointed sensors will forward point p to help increase the probability of transmitting point p successfully. In conclusion the proposed estimation of probability is a self-adapting function in the accordance with the status of the links and the routing structure of the different sensor networks, which tradeoffs the energy consumption and the accuracy of the query results well. In the following, we prove that the value of function $lpr(V_{i-1})_p$ is smaller than the probability that point p is sent to any sensor at the $i-1$th level by giving the following theorem.

Theorem 1. *Assume that point p is generated at sensor $v_{i',j'}$ and received by sensor $v_{i,j}$. Then, $Pr(V_{i-1})_p \geq lpr(V_{i-1})_p$.*

Proof. Recall that $Par(v_{i,j})_u$ is the set of partners of sensor $v_{i,j}$ from u, in which the sensors and $v_{i,j}$ have the same downstream sensor u. If sensor $v_{i,j}$ receives point p from some of its downstream sensors in $Down(v_{i,j})_{v_{i',j'}}$, it is obvious that the sensors in $\bigcup_{u \in Down(v_{i,j})_{v_{i',j'}}} Par(v_{i,j})_u$ are the ancestors of sensor $v_{i',j'}$ and likely to receive point p. For a partner $v \in \bigcup_{u \in Down(v_{i,j})_{v_{i',j'}}} Par(v_{i,j})_u$, the probability of that it received p is

$$pr(v)_p = 1 - \prod_{u \in Down(v)_{v_{i',j'}}} (1 - pr(u)_p)(1 - f)), \tag{4}$$

where v is at the ith level and each u in $Down(v)_{v_{i',j'}}$ is the downstream sensor sending point p to v. There may be a sensor $u \in Down(v)_{v_{i',j'}}$ that is not a downstream sensor of sensor $v_{i,j}$. Thus, set $Down(v)_{v_{i',j'}}$ and the value of $pr(u)_p$ in Eq. (4) is not known by sensor $v_{i,j}$, because every sensor only has the local information and consequently the value of $pr(V_{i-1})_p$ is not known either.

The probability that point p is sent to the upstream sensors of v is $pr(v)_p * (1 - f^{|Up(v)|})$. Thus, the probability of sending point p to any sensor in V_{i-1} is

$$pr(V_{i-1})_p = 1 - \prod_{v_{i,k} \in Anc(v_{i',j'})} (1 - pr(v_{i,k})_p(1 - f^{|Up(v_{i,k})|})), \tag{5}$$

where $v_{i,k}$ at the ith level is an ancestor of sensor $v_{i',j'}$.

Within Eq. (4), $pr(u)_p$ is the probability that p is sent from sensor $v_{i',j'}$ to a downstream sensor u of v at the $(i+1)$th level. The value of $pr(u)_p$ is minimum when point p is sent to sensor u through single path with $i' - i - 1$ hops, that is, $pr(u)_p \geq (1-f)^{i'-i-1}$. And $d(v)$ is the number of sensors in $Down(v)_{v_{i',j'}} \cap \{u_1, u_2, \ldots, u_m\}$, where $\{u_1, \ldots, u_m\}$ are the downstream sensors sending point p to $v_{i,j}$ successfully. Obviously $d(v) \leq Down(v)_{v_{i',j'}}$. Because $(1-f)^{i'-i-1} \leq pr(u)_p$ and $d(v) \leq |Down(v)_{v_{i',j'}}|$, from Eqs. (4) and (2), $lpr(v)_p \leq pr(v)_p$. $lpr(v)_p \leq pr(v)_p$ and $\{v_1, \ldots, v_{m'}\} \subseteq Anc(v_{i',j'})$, we have

$$\begin{aligned} lpr(V_{i-1})_p &= 1 - \prod_{t=1}^{m'} (1 - lpr(v_t)_p(1 - f^{|Up(v_t)|})) \\ &\leq 1 - \prod_{v_{i,k} \in Anc(v_{i',j'})} (1 - pr(v_{i,k})_p(1 - f^{|Up(v_{i,k})|})) \\ &= pr(V_{i-1})_p. \end{aligned} \tag{6}$$

Thus, if $lpr(V_{i-1})_p \geq \theta_p$, then $pr(V_{i-1})_p \geq \theta_p$.

Therefore, if a point p generated at $v_{i',j'}$ is forwarded to sensors at the ith level and there is a non-appointed sensor $v_{i,j}$ discarding point p because $lpr(V_{i-1})_p > \theta_p$, the probability that point p is sent to any sensor in V_{i-1} is also larger than θ_p. If a sensor $v_{i-1,k}$ receives point p and other k points with larger values than $p.val$, p is impossible to be part of top-k result and it is transmitted successfully; otherwise, if there is a sensor $v_{i-1,k}$ receiving point p but discarding it because $lpr(V_{i-1})_p \geq \theta_p$, the probability of sending point p to the sensors in V_{i-2} is not smaller than θ_p. Eventually point p is sent to the base station with probability not smaller than θ_p if there is at least a sensor at the second level discarding point p because the locally estimated probability is not smaller than θ_p.

3.4 The Extension of the Algorithm

It is well known that link failure rates in wireless sensor networks are not identical. Denote by $f_{u,v}$ the failure rate of a link from sensor u to sensor v. Our proposed algorithm can be extended for this generalized scenario as well. First, in local information collection phase, each sensor broadcasts the link failure rates of the links between itself and its upstream and downstream sensors in addition to the number links between them. Second, each sensor calculates the probabilities of receiving a point and forwarding the point along with the link failure probability. Finally, the local function is modified to suit for WSNs with different link failure rates. Denote by $pr(p)_v$ the probability of sensor v receiving point

p. If v is the generator of p, $pr(p)_v = 1$ and v broadcasts point p with $pr(p)_v$; otherwise, $pr(p)_v$ is calculated as follows. Assume that sensor v received point p with probabilities $pr(p)_{u_1}, \ldots, pr(p)_{u_m}$ from sensors $u_1, \ldots, u_m$. The probability that v receives p is $pr(p)_v = 1 - \prod_{i=1}^{m}(1 - pr(p)_{u_i} * f_{u_i,v})$. Therefore, sensor v estimates the probability that sensor v' in $\bigcup_{t=1}^{m} Par(v_{i,j})_{u_t}$ if it receives point p from sensor $u_1, u_2, \ldots, u_m$, which is

$$lpr(v')_p = 1 - \prod_{v' \in Up(u_t)} (1 - pr(p)_{u_t}(1 - f_{u_t,v'})), \tag{7}$$

where $1 \leq t \leq m$. $pr(p)_{u_t}$ is received by sensor v and $f_{u_t,v'}$ can be obtained through local information collection. Therefore the local function is modified as

$$lpr(V_{i-1})_p = 1 - \prod_{t=1}^{m'}(1 - lpr(v_t)_p(1 - \prod_{x=1}^{|Up(v_t)|} f_{v_t,n_x})), \tag{8}$$

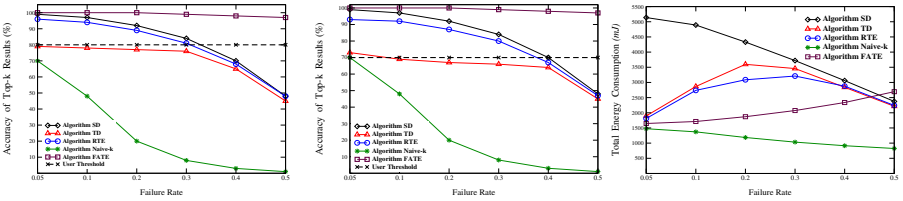
where $n_1, n_2, \ldots, n_x$ are the upstream sensors of v_t. All the variables in the modified function can be obtained by sensor v, and this indicates that the modified function can also be calculated by sensor v locally.

4 Performance Study

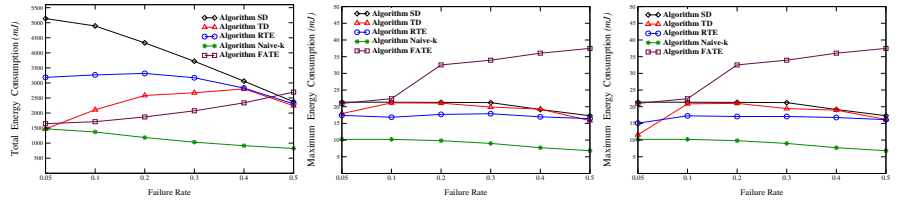
In this section we evaluate the performance of the proposed algorithm in terms of the total energy consumption, the maximum energy consumption among sensors, and the accuracy of the top-k query results. We assume that the sensor network is used to monitor a $100m \times 100m$ region of interest. Within the region, 1000 sensors are randomly deployed by the `NS-2` simulator [30] and the base station is located at the square center. There is a communication channel between two sensors if they are within the transmission range (5 meters in this paper) of each other. Each point is represented by 8 bytes. It is supposed that the energy overhead on transmitting and receiving a header and the handshaking are $\rho_t = 0.4608\ mJ$ and $\rho_r = 0.1152\ mJ$. The energy consumption of transmitting and receiving one byte are $R = 0.0144\ mJ$ and $r_e = 0.00576\ mJ$, respectively, following the parameters given in a commercial sensor MICA2 [28]. In our experiments, we use the real sensing dataset [29]. The performance of algorithm `SD` in [18], algorithm `TD` in [17], algorithm `FATE` based on the re-transmission mechanism in [12] and a well-known algorithm `Naive-`k in [22] on the tree structure is used as the benchmark for comparison purpose.

4.1 Performance Comparison with Equal Link Failure Rates

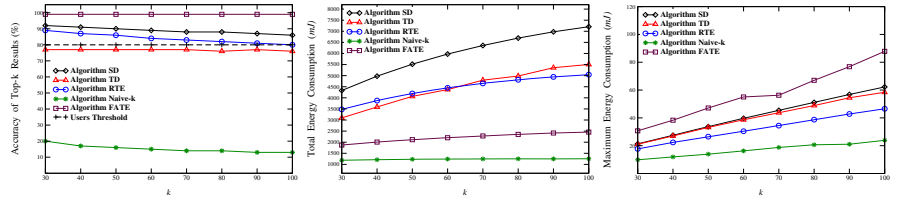
We first study the performance of various algorithms with different thresholds θ and equal link failure rates f, where θ is 0.7 or 0.8, while f is ranged from 0.05 to 0.5. Assume that a top-30 query is broadcast to all sensors. The results of the experiments are the average of running the top-30 query by different algorithms 1,000 times, and in each time the status of each link in the network is randomly determined, according to the given link failure rate.



(a) Accuracy of Top-k Results when θ=0.8 (b) Accuracy of Top-k Results when θ=0.7 (c) Total Energy Consumption when θ=0.8



(d) Total Energy Consumption when θ=0.7 (e) Max Energy Consumption when θ=0.8 (f) Max Energy Consumption when θ=0.7



(g) Accuracy of Top-k Results when θ=0.8, f = 0.2 (h) Total Energy Consumption when θ=0.8, f=0.2 (i) Max Energy Consumption when θ=0.8, f=0.2

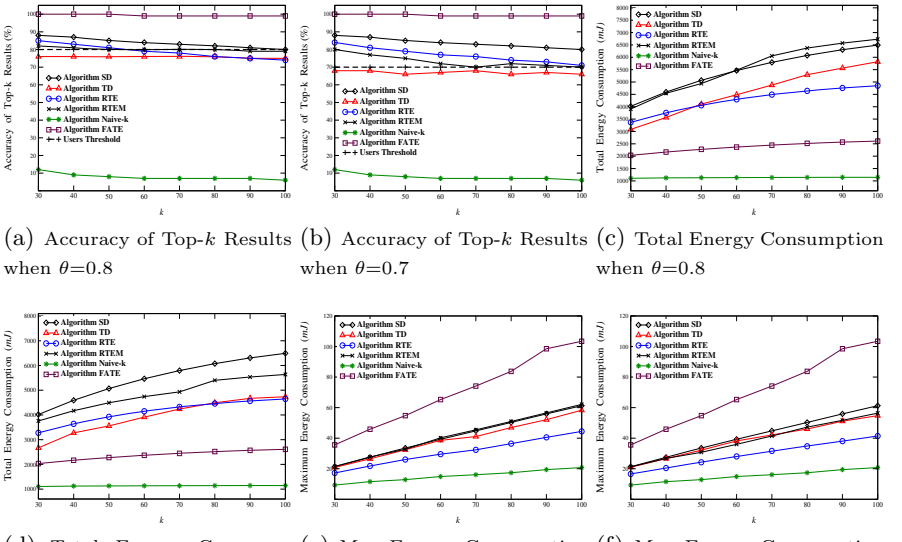
Fig. 2. The performance by various algorithms with identical link failure rates

From Fig. 2, we can observe that the query results delivered by algorithms `FATE` and `SD` has a higher accuracy than that of the other algorithms, but algorithm `FATE` has the largest maximum energy consumption and `SD` has the largest total energy consumption. The accuracy of query result by algorithm `Naive-`k drops sharply when the failure rate increases, which implies that it is not robust under the unstable communication environment. Compared to another adaptive algorithm `TD`, the results delivered by algorithm `RTE` is more accurate but with less energy consumption in overall. Figs. 2(g), 2(h), and 2(i) indicate the performance of various algorithms with different ks when θ = 0.8 and f = 0.2. It can be seen that the accuracy of query results by algorithm `RTE` is above the threshold and the total and the maximum energy consumptions by it is smaller than these by the other mentioned algorithms. This implies algorithm `RTE` makes a better trade off between the accuracy of the query result and the energy consumption.

4.2 Performance Comparison with Different Failure Rates

We then evaluate the performance of various algorithms with different link failure rates. We assume that the link failure rate of each link is randomly generated and with within the range from 0 to 0.5. The modified version of algorithm `RTE` for this general case is referred to as `RTEM`.

Fig. 3 illustrates the performance curves of different algorithms, in which we can see that with various values of θ, the query results delivered by algorithm `FATE` are the most accurate, while the accuracy of the results by algorithm `Naive-`k is the worst. As shown in Figs. 3(a) and 3(b), the accuracy of algorithm `RTEM` is above the broken line representing θ and they are closer to the curves of the accuracy of algorithm `SD` with the increase of k. Although there is no guarantees that the accuracy of query results by algorithms `RTE` and `TD` meets the given threshold, the query result accuracy by both algorithms is close to the threshold, due to the adaptive mechanisms embedding in both the algorithms. From Figs. 3(c) and 3(d), with the decrease of the threshold, the energy consumption by algorithm `RTEM` is close or smaller than that by algorithm `SD`, since the dynamic decision plays a crucial role in making the trade-off between the energy consumption and the accuracy of query results. And it is observed from Figs. 3(e) and 3(f) that algorithm `RTE` has the better performance in maximum energy consumption than any other algorithms except algorithm `Naive-`k.



(a) Accuracy of Top-k Results when θ=0.8

(b) Accuracy of Top-k Results when θ=0.7

(c) Total Energy Consumption when θ=0.8

(d) Total Energy Consumption when θ=0.7

(e) Max Energy Consumption when θ=0.8

(f) Max Energy Consumption when θ=0.7

Fig. 3. The performance by various algorithms with random link failure rates when θ ranges from 0.7 to 0.9

5 Conclusion

In this paper we have tackled the problem of robust top-k query processing in wireless sensor networks. The objective is to minimize the energy consumption with the accuracy of query results constraint. We proposed a localized algorithm to strive the finest tradeoff between the energy consumption and the accuracy of the results. We conducted extensive experiments by simulations to evaluate the performance of the proposed algorithm. The experimental results show that there is a non-trivial tradeoff between the energy consumption and the accuracy of query results. The proposed algorithm is more energy-efficient than that of the existing algorithms while meeting the specified accuracy requirement on query results.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine, 102–114 (2002)
2. Considine, J., Hadjieleftheriou, M., Li, F., Byers, J., Kollios, G.: Robust approximate aggregation in sensor data management systems. ACM TODS 34, 1–33 (2009)
3. Considine, J., Li, F., Kollios, G., Byers, J.: Approximate aggregation techniques for sensor databases. In: Proc. of ICDE, pp. 449–460. IEEE, Los Alamitos (2004)
4. Chen, B., Liang, W., Yu, J.X.: Progressive skyline query evaluation and maintenance in wireless sensor networks. In: Proc. of CIKM, pp. 1445–1448. ACM, New York (2009)
5. Chen, B., Liang, W., Yu, J.X.: Online time interval top-k queries in wireless sensor networks. In: Proc. of MDM, pp. 177–182. IEEE, Los Alamitos (2010)
6. Chen, B., Liang, W., Zhou, R., Yu, J.X.: Energy-efficient top-k query processing in wireless sensor networks. In: Proc. of CIKM, pp. 329–338. ACM, New York (2010)
7. Gobriel, S., Khattab, S., Mosse, D., Brustoloni, J., Melhem, R.: RideSharing: fault tolerant aggregation in sensor networks using corrective actions. In: IEEE SECON, pp. 595–604 (2006)
8. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Computing Survey 40(4) (2008)
9. Jurca, O., Michel, S., Herrmann, A., Aberer, K.: Continuous Query Evaluation over Distributed Sensor Networks. In: Proc. of ICDE, pp. 912–923. IEEE, Los Alamitos (2010)
10. Jiang, H., Cheng, J., Wang, D., Wang, C., Tan, G.: Continuous multi-dimensional top-k query processing in sensor networks. In: Proc. of INFOCOM. IEEE, Los Alamitos (2011)
11. Liang, W., Chen, B., Yu, J.X.: Top-k query evaluation in sensor networks under query response time constraint. Information Sciences 181, 869–882 (2011)
12. Lazaridis, I., Han, Q., Mehrotra, S., Venkatasubramanian, N.: Fault tolerant evaluation of continuous selection queries over sensor data. International Journal of Distributed Sensor Networks 5, 338–360 (2009)
13. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a tiny aggregation service for ad hoc sensor networks. ACM SIGOPS Operating Systems Review 36, 131–146 (2002)

14. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: The design of an acquisitional query processor for sensor networks. In: Proc. of SIGMOD, pp. 491–502. ACM, New York (2003)
15. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TinyDB: An acquisitional query processing system for sensor networks. ACM Trans. Database Systems 30, 122–173 (2005)
16. Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proc. of International Workshop on Wireless Sensor Networks and Applications, pp. 88–97. ACM, New York (2002)
17. Manjhi, A., Nath, S., Gibbonsgh, P.B.: Tributaries and deltas: efficient and robust aggregation in sensor network streams. In: Proc. of SIGMOD, pp. 287–298. ACM, New York (2005)
18. Nath, S., Gibbons, P.B., Seshan, S., Anderson, Z.R.: Synopsis diffusion for robust aggregation in sensor networks. In: Proc. of SenSys, pp. 250–262. ACM, New York (2004)
19. Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. Communication of the ACM 43(5), 51–58 (2000)
20. Stern, M., Buchmann, E., Bohm, K.: Towards efficient processing of general-purpose joins in sensor networks. In: Proc. of ICDE, pp. 126–138. IEEE, Los Alamitos (2009)
21. Stern, M., Bohm, K., Buchmann, E.: Processing continuous join queries in sensor networks: a filtering approach. In: Proc. of SIGMOD, pp. 267–278. ACM, New York (2010)
22. Silberstein, A., Braynard, R., Ellis, C., Munagala, K., Yang, J.: A sampling-based approach to optimizing top-k queries in sensor networks. In: Proc. of ICDE, pp. 68–79. IEEE, Los Alamitos (2006)
23. Wu, M., Xu, J., Tang, X., Lee, W.-C.: Top-k monitoring in wireless sensor networks. IEEE Trans. Knowledge and Data Engineering 19(7), 962–976 (2007)
24. Yao, Y., Gehrke, J.: The cougar approach to in-network query processing in sensor networks. ACM SIGMOD Record 31, 9–18 (2002)
25. Yao, Y., Gehrke, J.: Query processing for sensor networks. In: Proc. of CIDR (January 2003)
26. Ye, F., Zhong, G., Lu, S., Zhang, L.: GRAdient broadcast: a robust data delivery protocol for large scale sensor networks. Wireless Networks 11, 285–298 (2005)
27. Zhao, J., Govindan, R.: Understanding packet delivery performance in dense wireless sensor networks. In: Proc. of SenSys, pp. 1–13. ACM, New York (2003)
28. Crossbow Inc.: MPR-Mote Processor Radio Board Users Manual
29. http://db.csail.mit.edu/labdata/labdata.html (2004)
30. The Network Simulator-ns2 (2006), http://www.isi.edu/nsnam/ns

Learning Top-k Transformation Rules*

Sunanda Patro and Wei Wang

University of New South Wales
{sunandap,weiw}@cse.unsw.edu.au

Abstract. Record linkage identifies multiple records referring to the same entity even if they are not bit-wise identical. It is thus an essential technology for data integration and data cleansing. Existing record linkage approaches are mainly relying on similarity functions based on the surface forms of the records, and hence are not able to identify complex coreference records. This seriously limits the effectiveness of existing approaches.

In this work, we propose an automatic method to extract top-k high quality *transformation rules* given a set of possibly coreferent record pairs. We propose an effective algorithm that performs careful local analyses for each record pair and generates candidate rules; the algorithm finally chooses top-k rules based on a scoring function. We have conducted extensive experiments on real datasets, and our proposed algorithm has substantial advantage over the previous algorithm in both effectiveness and efficiency.

1 Introduction

Real data are inevitably noisy, inconsistent, or contain errors. For example, there are *dozens* of correct ways to cite a publication, depending on the bibliography style one uses; however, there can be hundreds of citations to the same publication that contain errors, as caused by typographical errors, Optical Character Recognition (OCR) errors, or errors introduced when the citation was extracted by a program from Web pages.

Record linkage is the process of bringing together two or more separate records pertaining to the same entity, even if their surface forms are different. It is a cornerstone to ensure high quality of mission-critical data either in a single database or during data integration from multiple data sources. Therefore, it has been used in many applications including data cleansing, data integration from multiple sources or the Web, etc.

Most existing methods for record linkage rely on similarity functions to generate candidate pair of records that may be coreferent. This is insufficient when coreferent records has little surface similarity. For example, `23rd` and `twenty-third`, `VLDB` and `Very Large Databases`. Therefore, existing systems

* This work was partially supported by ARC Discovery Projects DP0987273 and DP0881779.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 172–186, 2011.

incorporate *transformation rules* to recognize these domain-specific equivalence relationships.

Traditionally, transformation rules were manually created by experts. This process is not only tedious, expensive, and often erroneous, the generated rules are seldom comprehensive enough. This is the main motivation for semi-automatic methods to automatically learn a set of high quality candidate rules [1]; domain experts can then manually validate or refine the candidate rules. Hence, it is important that a majority of the candidate rules generated by these algorithms should be correct. The rule learning algorithm should also be able to cope with large input datasets and learn top-k rules efficiently. As we demonstrate in the experiments, existing methods [1] fail to meet both requirements.

In this paper, we propose a novel method to automatically learn top-k transformation rules from a set of input record pairs known to be coreferent. We perform meticulous local alignment for each pair of records by considering a set of commonly used edit operations. We then generate a number of candidate rules based on the optimal local alignment. Statistics of the candidate rules are maintained and aggregated to select the final top-k rules. We have conducted extensive experiments with the existing state-of-the-art algorithm. We found that our rule learning algorithm outperforms the existing method in both effectiveness and efficiency.

Our contributions can be summarized as:

- We proposed a local alignment-based rule learning algorithm. Compared with the global greedy algorithm in [1], our algorithm generates fewer candidate rules, and our candidate rules are more likely to be correct rules.
- We have performed extensive experiments using several publicly available real-world datasets; our experimental results shows a 3.3x increase in the percentage of correct rules as compared with previous approach, and up to 300x speed-up in efficiency.

The rest of the paper is organized as follows: Section 2 introduces related work. We present our algorithm in Section 3. Experimental results are given in Section 4, and Section 5 concludes the paper.

2 Related

Record linkage is known under many different names, including entity resolution, and near duplicate detection. It is a well studied problem and has accumulated a large body of work. We refer readers to surveys [2,3], and focus on related work that is most closely related to our proposal.

Most existing record linkage approaches exploit similarities between values of intrinsic attributes. Many similarity or distance functions have been proposed to model different types of errors. For example, edit distance is used to account for typographical errors and misspellings [4]. Jaro-Winkler distance is designed for comparing English names [5]. Soundex is used to account for misspellings due to similar pronunciations. Jaccard similarities or cosine similarities measure the similarity of multi-token strings [6]. Similarity functions can also be

weighted. A common heuristic is to use the well-known *tf-idf* scheme. To allow misspellings, a soft version of *tf-idf* is proposed in [7], and later extended in [8]. Alternatively, weights can be learned automatically using machine learning techniques [9,10,11]. Multiple similarity functions can be used on the same or different attributes of the records. The similarity values can either be simply aggregated or be treated as a relational input to a classifier [10].

Dealing with Complex Coreferent Records. It is well-known that a single similarity function is not sufficient to identify complex coreferent records. Past efforts can be categorized into several categories below.

The first category is to learn a good similarity function using machine learning techniques [12,13,11]. For example, [14] employs a two-phase approach. In the first phase, attribute value matchers are tuned, and in the second phase, a SVM classifier is learned from a combination of the tuned matchers generated from the previous step. The experimental results show that trainable similarity measures are capable of learning the specific notion of similarity that is appropriate for a specific domain. While this approach focuses on homogenous string-based transformation, [15] uses heterogeneous set of models to relate complex domain specific relationships between two values. One technical issue for these learning-based approach is the selection of training datasets, especially the negative instances. [10] employs active learning techniques to minimize the interaction with users, and is recently improved by [16].

Another category of approaches is to model complex or domain-specific transformation rules [17,1]. We compare with them in more detail in the next subsection. The learned rules can be used to identify more coreferent pairs [18,19].

Note that although only few works in record linkage focus on transformation rules, they have been widely employed in many other areas, and automatic rule learning algorithms have been developed accordingly. In Natural Language Processing (NLP), [20] finds sets of synonyms by considering word co-occurrences. Later, [21] utilizes the similar idea to identify paraphrases and grammatical sentences by looking at co-occurrence of set of words.

Recently, researchers have used transformation rules to deduplicate URLs without even fetching the content [22,23], e.g., `http://en.wikipedia.org/?title=*` and `http://en.wikipedia.org/wiki/*` always refer to the same web page. However, the rules and their discovery algorithms are heavily tailored for URLs.

Another closely related area is the substitution rules used in *query rewriting* [24,25]. For example, when a user submits a query `apple music player` to a search engine, it may change the query to `apple ipod`. The substitution rules are mainly mined from query logs, and the key challenges are how to find similar queries and how to rank them.

Existing Transformation Rule Learner. [1] is a recent work to learn top-k transformation rules given a set of coreferent record pairs. it has shown much better accuracy and scalability to larger datasets than the previous approach [17]. The idea of [1] is to identify candidate rules from the unmapped tokens of each

record pair, and then compute the aggregated score of the candidate rules to find the top-k high quality rules. While our proposal follows the similar framework, there are a few major differences:

- We perform more refined local alignment for each record pair. We use a set of common edit operations (including typographical errors and abbreviations) to identify more mappings tokens, and hence produce fewer number of candidate rules from unmapped tokens. E.g., in Figure 1, all yellow and green tokens will be unmapped in [1]'s approach, and candidate rules which are essentially all possible mapping among them will be generated and counted. [1]'s strategy of pairing of all possible subsets of unmapped tokens not only decreases the quality of the final top-k rules, but also slows down its rule learning speed substantially.
- We can optionally support partitioning the records into fields (i.e., partitions), which further reduces the number of candidate rules, and speeds up the computation.
- We use a better scoring function than [1], which is less impacted by the length of the rules and counts the frequency of the rules by the clusters they appeared.

As demonstrated in our experiments (Section 4), these differences result in substantial improvements of our proposed algorithm over [1] in both the effectiveness and efficiency.

3 The Local-Alignment-Based Algorithm

Similar to [1], the input to transformation rule learning algorithms is a set of coreferent record pairs. The overall idea of our new algorithm is to perform careful *local alignment* for each record pair first, generate *candidate rules* from the optimal local alignment, and aggregate the "strength" of the rules over all input pairs to winnow high-quality rules from all the candidate rules.

3.1 Preprocessing the Input Data

We perform the standard preprocessing for input record pairs: we remove all non-alphanumeric characters except spaces, and then converting characters to lower cases. We do not use the case information as it is not reliable for noisy input data. We then tokenize the strings into a sequence of tokens using white space as the separator. We also remove stopwords.

3.2 Segmentation

The goal of segmentation is to decompose a record into a set of semantic constituents known as *fields*, i.e., a substring of tokens in a tokenized record. For example, bibliography records can usually be segmented into the `authors`, `title`, and `venue` fields. Our framework does not rely on a specific type of segmentation method. One can use either supervised methods (e.g., CRF [26]) or simple rule-based segmentation methods (e.g., based on punctuations in the raw input records).

Although this step is optional, appropriate segmentation is beneficial to generating better rules and faster execution of the algorithm. This is mainly because (i) We do not consider mappings for two tokens in different fields in two records. Hence fewer candidate rules are generated, and this speeds up the algorithm. (ii) Most of the rules generated across fields are actually erroneous. (iii) It is possible that we can use different parameter settings to learn rules for a particular field. For example, transformation rules for author names (e.g., omitting the middle name) probably do not apply on paper titles. We do not explore this option in this work and leave it for future work.

3.3 Local Alignment

We perform field alignment and then find the optimal local alignment between the values of the corresponding fields.

Computing the Optimal Local Alignment

We need to find a series of edit operations with the least cost to transform one string to another. We analyzed common transformations and decided that we support the following set of common edit operations:

- **Copy:** copy the token exactly.
- **Abbreviation:** allows one token to be a *subsequence* of another token, e.g., `department` ⇔ `dept`.
- **Initial:** allows one single-letter token to be equivalent to the first letter of another token, e.g., `peter` ⇔ `p`.
- **Edit:** allows the usual edit operations (i.e., Insertion, Deletion, and Substitution), e.g., `schütze` ⇔ `schuetze`
- **Unmapped:** tokens that are not involved in any transformation are denoted as *unmapped* tokens.

In addition to assigning cost to each of the above edit operations, we also prioritize them as *copy* > *initial* > *abbreviation* > *edit* > *unmapped*. In other words, if a high priority operation can convert one token to another, we do not consider operations of lower priorities. For example, between two tokens `department` and `dept`, since an *abbreviation* operation can change one into another, we do not consider *edit* operations. The cost of *copy* is always 0, and the cost of *unmapped* is always higher than other costs; the cost of other operations are subject to tuning. Algorithm 1 performs such local alignment and returns the minimum cost between two strings.

Example 1. We show two strings S and T (i.e., there is no segmentation), and the optimal local alignment in Figure 1.

Aligning Fields

If the input records have been partitioned into fields, we also need to align the fields. In the easy case where each field has a class label (as those output by the supervised segmentation methods), the alignment of fields is trivial — we just

Algorithm 1. AlignStrings(S, T)

```
1 c ← 0;
2 for each token w ∈ S do
3     mincost ← min{cost(w, w') | w' ∈ T};
      /* follows the priorities of edit operations                    */
4     if w is not unmapped then
5         c ← c + mincost;
6 for each unmapped token w ∈ S ∪ T do
7     c ← c + unmapped_cost(w);
8 return c
```

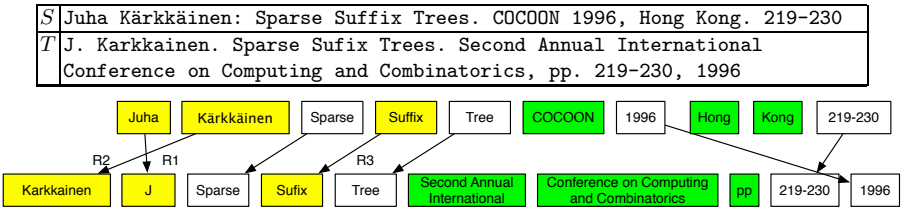


Fig. 1. Optimal Local Alignment (White cells are copied, green cells are unmapped, and yellow cells involve other types of edit operations)

align fields with the same class label (e.g., `authors` to `authors`). If fields do not have class labels, we use Algorithm 2 to find the optimal alignment — an alignment such that the total cost is minimum. This is done by reducing the problem into a maximum weighted bipartite graph matching problem, which can be efficiently solved by the Hungarian algorithm [27] in $O(V^2 \log V + VE) = O(B_{max}^3)$, where B_{max} is the maximum number of fields in the records.

Algorithm 2. AlignBlocks(X, Y)

```
1 Construct a weighted bipartite graph G = (A ∪ B, E, λ), such that there is a 1-to-
  1 mapping between nodes in A and the fields in X, and there is a 1-to-1 mapping
  between nodes in B and fields in Y, E = {e_ij} connects A_i and B_j, and the
  weight of an edge is the negative of its cost, i.e., λ(e_ij) = −AlignStrings(A_i, B_j);
2 M = FindMaxMatching(G) ;                    /* use the Hungarian alg */
3 return M
```

Multi-token Abbreviation

Some of the tokens are unmapped because they involve in *multi-token abbreviation*. We generalize the *abbreviation* operation to include multiple-token to one token abbreviation. For example, in Figure 1, one such instance is `Conference on Computing and Combinatorics` to `cocoon`.

We classify multi-token abbreviations into the following categories:

- **Acronym:** A set of tokens $[u_1, u_2, \ldots, u_k]$ maps to a token v via an acronym mapping, if there exists a sequence of prefix length l_i for each u_i, such that $u_1[1, l_1] \circ u_2[1, l_2] \circ \ldots u_k[1, l_k] = v$, where $\circ$ concatenates two strings. E.g., association computing machinery $\Leftrightarrow$ acm.
- **Partial Acronym:** A set of tokens $[u_1, u_2, \ldots, u_k]$ maps to a token v via a partial acronym mapping, if there exists a prefix or suffix, v_s, of v longer than a minimum length threshold, and a subsequence $ss(u_i)$ of for each u_i, such that $ss(u_1) \circ ss(u_2) \circ \ldots ss(u_k) = v_s$. In other words, v_s is equal to a subsequence of the concatenated string $u_1 \circ \ldots \circ u_k$. E.g., conference on knowledge discovery and data mining $\Leftrightarrow$ sigkdd.

Algorithm 3. MultiToken-Abbreviation(X, Y)

```
Input : X and Y are the input record pairs
for each remaining unmapped token v ∈ X do
    for each remaining contiguous set of unmapped tokens u in Y do
        str ← u_1 ∘ ... ∘ u_k;
        if exists a prefix or suffix, v_s, of v such that v_s is a subsequence of str then
            partialAcronym ← true;
            if each of the match in u_i starts from its first character then
                fullAcronym ← true;
        if partialAcronym = true or fullAcronym = true then
            remove matched tokens from X and Y;
```

The algorithm to discover both types of acronyms is depicted in Algorithm 3.

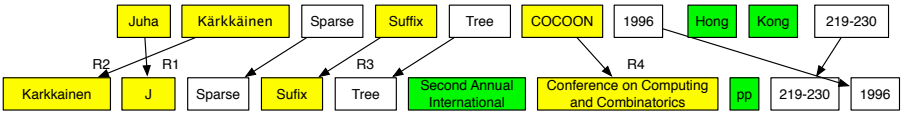


Fig. 2. Optimal Local Alignment After Recognizing Multi-token Abbreviation (R_4)

Example 2. As illustrated in Figure 2, the partial acronym mapping, denoted by R_4, is recognized.

3.4 Obtaining Top-k Rules

We first generate candidate rules, compute and aggregate their scores across all input pairs of records, and finally select the top-k rules.

Definition 1 (Rule). *A rule is in the form of* `lhs` ⇔ `rhs`*, where both* `lhs` *and* `rhs` *are a sequence of tokens. The rule means that* `lhs` *is equivalent to* `rhs`*.*

Definition 2 (Atomic Rule). *An atomic rule is a rule where either the* `lhs` *or the* `rhs` *is a single token or empty (denoted as* ⊥*).*

Note that an *omission* rule is a special rule where one side of the rule is empty. E.g., since `pp` is still unmapped in Figure 2, we have an omission rule `pp` ⇔ ⊥.

Depending on the number of tokens on each side of the rule, we can have 1-1 rule (e.g., `peter` ⇔ `p`), 1-n rule (e.g., `vldb` ⇔ `very large databases`), n-m rule (e.g., `information systems` ⇔ `inf sys`). In practice, we observe that almost all n-m rules can be decomposed into several 1-1 or 1-n rules. E.g., the n-m rule above can be decomposed into two 1-n rules: `information` ⇔ `inf` and `systems` ⇔ `sys`. Therefore, we focus on finding atomic rules and assemble them to find more n-m rules.

Generating Candidate Rules

At this stage, we can generate rules from mapped tokens and unmapped tokens in different manners:

- For mappings (identified either by local alignment (Algorithm 1) or by multi-token abbreviation (Algorithm 3)), we can easily generate the rules. Note that we do not generate trivial copying rules (i.e., `A` ⇔ `A`). We also combine adjacent rules into n-m rules. E.g., once we identify two mappings: `information` ⇔ `inf` and `systems` ⇔ `sys`, and if `information` and `systems` are adjacent and `inf` and `sys` are adjacent too, then we also generate the rule `information systems` ⇔ `inf sys`.
- For each of the remaining unmapped tokens in one string, we postulate that it might be deleted (i.e., transformed to ⊥), or it is mapped to every contiguous subset of the unmapped tokens in the other string. We do this for both strings. It is expected that although many of these candidate rules are invalid rules, some valid rules (usually corresponds to complex, domain-specific transformations sharing little surface similarities[1]) will appear frequently if we aggregate over large amount of input pairs.

Example 3. Based on the mappings in Figure 2, we generate the following rules from mapped tokens:

- `Juha` ⇔ `J`,
- `Käkkäinen` ⇔ `Kakkainen`,
- `Juha Käkkäinen` ⇔ `Kakkainen J`,
- `Suffix` ⇔ `Sufix`,
- `COCCON` ⇔ `Conference on Computing and Combinatorics`.

Candidate rules generated from unmapped tokens `pp` are:

- `pp` ⇔ ⊥
- `pp` ⇔ `Hong`

[1] One example is `maaliskuu` ⇔ `march` found in our experiments, where `maaliskuu` in Finnish means `march` in English.

– pp ⇔ Kong
– pp ⇔ Hong Kong

Score of a Rule

Definition 3. *We define the score of a rule R as:*

$$score(R) = \log(1 + freq(R)) \cdot \log(1 + len(R)) \cdot wt(R), \tag{1}$$

where $freq(R)$ is the number of occurrences of the rule R, $len(R)$ is the total number of tokens in R, and $wt(R)$ is the weight assigned based on the type of the rule.

The above heuristic definition takes into consideration of the popularity of a rule its length, and its type. The length component is important because whenever ir ⇔ information retrieval holds, ir ⇔ information also holds; thus *score*(ir ⇔ information retrieval) ≤ *score*(ir ⇔ information), and the partial correct rule will be incorrectly ranked higher than the complete rule. The rule type component is used to capture the intuition that, e.g., a rule generated by mapped tokens is more plausible than that generated by (randomly) pairing unmapped tokens.

Select the top-k Rules

The complete algorithm is given in Algorithm 4 which learns top-k rules with the maximum scores from a set of input pairs of records.

In Algorithm 4, Lines 1–11 generate all the candidate rules from the mappings obtained for each record pair by considering the best local alignment and possible multi-token abbreviation. Lines 12–13 calculate the scores for each candidate rule. Then we iteratively select the *TopRule* which has the maximum score (Lines 16–20), and withdraw support from other conflicting candidate rules by the procedure UpdateRules. We repeat this process until k high-quality rules are found.

The procedure UpdateRules is illustrated in Algorithm 5. We say a rule R_1 conflicts with another rule R_2 if and only if one side of R_1 is identical to one side of R_2. Two kinds of typical conflicting rules are:

– A ⇔ B and A ⇔ C
– A ⇔ BCD and A ⇔ B

Analysis of the Algorithm. Let the number of input record pairs be N, the average number of candidate rules generated by each record pair be f. Then, $|\mathcal{R}| = f \cdot N$. The time complexity of Algorithm 4 is $O(k \cdot (|\mathcal{R}| + N \cdot f)) = O(k|\mathcal{R}|) = O(k \cdot N \cdot f)$. In practice, we observed that only a constant number of rules will be in conflict with the *TopRule* in each of the k iteration. Hence the time complexity is expected to be $O(k \cdot N)$.

Algorithm 4. Top$k$Rule($\mathcal{D}$, k)

Input : $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ are N input record pairs.

$\mathcal{R} \leftarrow \emptyset$;
for each *input record pair* $p_i = (X, Y)$ **do**
  $b_X \leftarrow$ segment X into partitions;
  $b_Y \leftarrow$ segment Y into partitions;
  $M \leftarrow$ AlignBlocks(b_X, b_Y);
  for each *candidate rule* R *generated from the mapping* M **do**
    if $R \notin \mathcal{R}$ **then**
      $R.support = \{p_i\}$;
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{R\}$;
    else
      Update the $R \in \mathcal{R}$ so that $R.support$ now includes p_i;
for each *candidate rule* $R \in \mathcal{R}$ **do**
  $R.score \leftarrow$ calcScore($R.support$) ; /* based on Equation (1) */
output $\leftarrow \emptyset$;
$i \leftarrow 1$;
while $i \leq k$ **do**
  $TopRule \leftarrow \arg\max_{R \in \mathcal{R}} \{R.score\}$;
  UpdateRules($\mathcal{R}$, $TopRule$);
  output $\leftarrow$ *output* $\cup$ $TopRule$;
  $i \leftarrow i + 1$;
return *output*

Algorithm 5. UpdateRules($\mathcal{R}$, $TopRule$)

$\{p_1, p_2, \ldots, p_l\} \leftarrow$ the support of rule $TopRule$;
for each p_i **do**
  $CR \leftarrow$ all the rules that conflict with $TopRule$ from the record pair p_i;
  for each *rule* $R \in CR$ **do**
    $R.support \leftarrow R.support \setminus \{p_i\}$;

4 Experiment

In this section, we perform experimental evaluations and analyses.

4.1 Experiment Setup

We use the following algorithms in the experiments:

Greedy. This is the state-of-the-art algorithm for learning top-k transformation rules by Microsoft researchers [1].

LA. This is our proposed local-alignment-based algorithm.

Note that for our LA algorithm, the weights set for various types of rules in Equation (1) are: rules from mapped tokens: 4, exact acronym: 3, partial acronym: 2, and others: 1. We use a partitioning method to segment each record into three fields based on an algorithm that takes into consideration the local alignment costs. In the interest of space, we will leave it to the full version of the paper.

All experiments were performed on a PC with AMD Opteron Processor 8378 CPU and 96GB memory, running Linux 2.6.32. All programs are implemented in Java.

We use the following three real-world datasets.

CCSB. This dataset comes from the Collection of Computer Science Bibliographies.[2] We queried the site with 38 keyword queries (e.g., `data integration`), and collected the top-200 results. Each query result is referred to a *cluster*, as it may contain multiple citations to the same paper. We only kept those clusters whose size is larger than one. We used five different bibliography styles.[3] We applied the i-th bibliography style to the i-th citations (if any) in each cluster, and got the corresponding transformed string from the LaTeX output. As a result, 3030 clusters were generated. We then form all possible pairs of the strings produced by LaTeX within the same cluster, and use them as the input record pairs for the transformation rule learning algorithms. This results in 12,456 pairs.

Cora. This is the hand-labeled Cora dataset from the RIDDLE project.[4] It contains 1,295 citations of 112 Computer Science papers. We clustered the citations according to the actual paper they refer to, and then generated all possible pairs within each cluster. As a result, we had 112 clusters with a total of 17,184 input record pairs.

Restaurant. This is the Restaurant dataset from the RIDDLE project. It contains 533 and 331 restaurants assembled from Fodor's and Zagat's restaurant guides, and 112 pairs of coreferent restaurants were identified. We applied similar record pair generation method as above, and generated 112 clusters and a total of 112 record pairs.

Note that we generate all pairs of citations in the same cluster (i.e., referring to the same publication) to exploit the ground truth data maximally. This, however, introduces a bias into the frequency of the rules (i.e., $freq(R)$ in Equation (1)). For example, for a cluster of size $2t$, a rule `A` $\Leftrightarrow$ `B` can have a frequency of up to t^2 from this cluster alone. To remedy this problem, we define the frequency as the number of *clusters* (rather than *record pairs*) such that the rule is generated. This is applied to both algorithms.

[2] `http://liinwww.ira.uka.de/bibliography/Misc/CiteSeer/`

[3] They are `these`, `acm`, `finplain`, `abbrv`, and `naturemag`, mainly from `http://amath.colorado.edu/documentation/LaTeX/reference/faq/bibstyles.pdf`.

[4] `http://www.cs.utexas.edu/users/ml/riddle/data.html`

4.2 Quality of the Rules

For both the Greedy and LA algorithms, we generate top-k rules for each dataset with varying number of k. We then validate *all* the output rules which were classified by domain experts into one of the three categories:

Correct when the rule is absolutely correct. E.g., `proceedings` ⇔ `proc`.
Partially Correct when the rule is partially correct. E.g., `ipl information processing letters` ⇔ `inf process lett vol`.
Incorrect when the rule is absolutely incorrect. E.g., `computer science` ⇔ `volume`.

Some of the correct rules discovered by LA are shown in Table 1.

Table 1. Example Rules Found

ID	Rule
1	`focs` ⇔ `annual ieee symposium on foundations of computer science`
2	`computer science` ⇔ `comput sci`
3	`pages` ⇔ `pp`
4	`5th` ⇔ `fifth`
5	`maaliskuu` ⇔ `march`

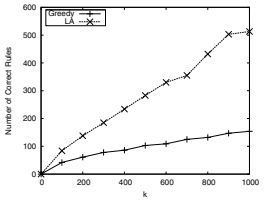
To evaluate the quality of rules generated by the algorithms, we count the number of *correct* and *incorrect* rules. We also calculate *precision* as the fraction of correct rules in the top-k output rules.[5] Note that we essentially ignore partially correct rules.

Figures 3(a), 3(b), and 3(c) show the numbers of correct rules for two algorithms by varying k on three datasets. Figures 3(d), 3(e), and 3(f) show the number of incorrect rules by varying k. We can see that

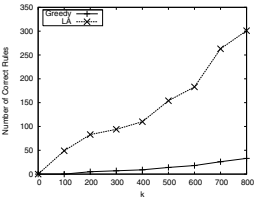
- LA outperforms Greedy substantially on all datasets by generating more correct rules than incorrect rules.
- Since the Cora dataset is dirtier than the CCSB dataset, the precision of both algorithms is lower. It can also been seen that the Greedy algorithm is affected more by the noise in the dataset than the LA algorithm.
- Since the Restaurant dataset is small, only fewer rules are generated. For the top-50 rules, LA generate 68% correct rules whereas Greedy has a precision of 34%.

We also plot the precision vs. k on the CCSB dataset in Figure 3(g). Since both algorithms strive to find high-quality rules first, the precision is highest when k is small, and decreases with increasing k. Precisions for both algorithms become stable for $k \geq 500$. In all settings, LA's precision is much higher than Greedy's. As we can see from the figure, the precisions of LA and Greedy is 84% and 42% when $k = 100$, and 51.3% and 15.4% when $k = 1000$.

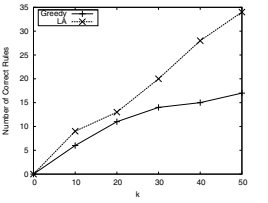
[5] Note that this definition is different from that in [1], where precision was defined as the number of incorrect rules.



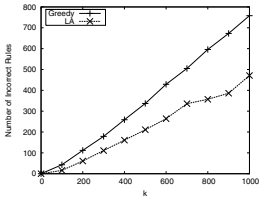
(a) CCSB, Number of Correct Rules



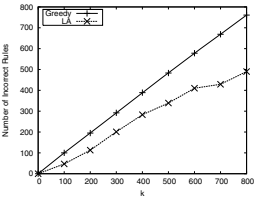
(b) Cora, Number of Correct Rules



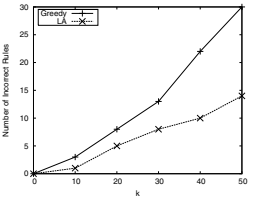
(c) Restaurant, Number of Correct Rules



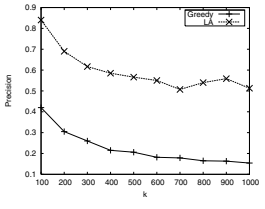
(d) CCSB, Number of Incorrect Rules



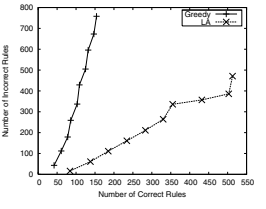
(e) Cora, Number of Incorrect Rules



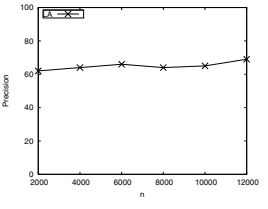
(f) Restaurant, Number of Incorrect Rules



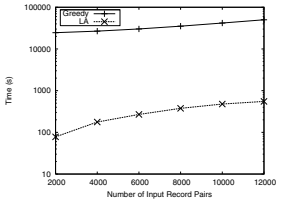
(g) CCSB, Precision vs. k



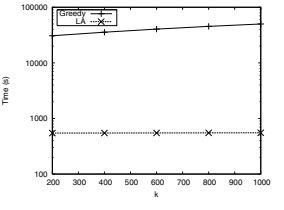
(h) CCSB, Number of Incorrect Rules vs. Number of Correct Rules



(i) CCSB, Precision vs. n



(j) CCSB, Time vs. n ($k = 1000$)



(k) CCSB, Time vs. k ($n = 12,000$)

Fig. 3. Experimental Results

Figure 3(h) shows how many incorrect rules are generated for a given target of correct rules on the CCSB dataset, as [1] did. It can be seen that much more incorrect rules are generated by the Greedy algorithm than the LA algorithm for any fixed amount of correct rules.

A good transformation rule learner should perform consistently when we vary the number of input record pairs. Such results are shown in Figure 3(i), where the input number of record pairs vary from 2,000 to 12,000, and $k = 200$. It can be observed that the precision of our LA algorithm remains stable (between 62% to 69%).

4.3 Execution Time

We investigate the efficiency of the algorithms versus the number of input record pairs (n). We measure the running time of the algorithms. The result is shown in Figure 3(j). We can see that the time grows more quickly for the Greedy algorithm than the LA algorithm. This is mainly because with the increasing input pairs, there are many more candidate rules generated by the Greedy algorithm as they do not perform careful local alignment. The running time of Greedy is up to 300 times more than LA also because of the vast amount of candidate rules generated.

We plot the running time versus the output size k in Figure 3(k). Both algorithms require more time when k increases, but Greedy's time grows quickly with k. This is mainly because Greedy needs to update the support of the vast amount of candidate rules in each iteration and hence takes much more time.

5 Conclusions

In this paper, we propose an effective and efficient top-k transformation rule learning algorithm. The algorithm is based on performing careful local alignment of input coreferent record pairs, and generating candidate rules based on the optimal local alignment. Our experiments demonstrate that our method generates more correct rules than the global greedy approach [1] in less amount of time.

References

1. Arasu, A., Chaudhuri, S., Kaushik, R.: Learning string transformations from examples. PVLDB 2, 514–525 (2009)
2. Winkler, W.E.: The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau (1999)
3. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. Knowl. Data Eng. 19, 1–16 (2007)
4. Wang, W., Xiao, C., Lin, X., Zhang, C.: Efficient approximate entity extraction with edit distance constraints. In: SIGMOD (2009)
5. Winkler, W.E., Thibaudeau, Y.: An application of the Fellegi-Sunter model of record linkage to the 1990 U.S. Decennial Census. Technical report, US Bureau of the Census (1991)
6. Xiao, C., Wang, W., Lin, X., Yu, J.X.: Efficient similarity joins for near duplicate detection. In: WWW (2008)
7. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string metrics for matching names and records. In: Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (2003)
8. Moreau, E., Yvon, F., Cappé, O.: Robust similarity measures for named entities matching. In: COLING (2008)
9. Bilenko, M., Mooney, R.J.: Learning to combine trained distance metrics for duplicate detection in databases. Technical report, University of Texas at Austin (2002)
10. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: KDD, pp. 269–278 (2002)

11. Tejada, S., Knoblock, C.A., Minton, S.: Learning domain-independent string transformation weights for high accuracy object identification. In: KDD, pp. 350–359 (2002)
12. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: KDD, pp. 475–480 (2002)
13. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD, pp. 39–48 (2003)
14. Bilenko, M., Mooney, R.J.: On evaluation and training-set construction for duplicate detection. In: Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation, Washington, DC, pp. 7–12 (2003)
15. Minton, S., Nanjo, C., Knoblock, C.A., Michalowski, M., Michelson, M.: A heterogeneous field matching method for record linkage. In: ICDM, pp. 314–321 (2005)
16. Arasu, A., Götz, M., Kaushik, R.: On active learning of record matching packages. In: SIGMOD Conference, pp. 783–794 (2010)
17. Michelson, M., Knoblock, C.A.: Mining the heterogeneous transformations between data sources to aid record linkage. In: IC-AI (2009)
18. Arasu, A., Chaudhuri, S., Kaushik, R.: Transformation-based framework for record matching. In: ICDE, pp. 40–49 (2008)
19. Arasu, A., Chaudhuri, S., Ganjam, K., Kaushik, R.: Incorporating string transformations in record matching. In: SIGMOD Conference, pp. 1231–1234 (2008)
20. Turney, P.D.: Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 491–502. Springer, Heidelberg (2001)
21. Pang, B., Knight, K., Marcu, D.: Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In: HLT-NAACL (2003)
22. Bar-Yossef, Z., Keidar, I., Schonfeld, U.: Do not crawl in the dust: different urls with similar text. In: WWW (2007)
23. Dasgupta, A., Kumar, R., Sasturkar, A.: De-duping urls via rewrite rules. In: KDD, pp. 186–194 (2008)
24. Jones, R., Rey, B., Madani, O., Greiner, W.: Generating query substitutions. In: WWW (2006)
25. Radlinski, F., Broder, A.Z., Ciccolo, P., Gabrilovich, E., Josifovski, V., Riedel, L.: Optimizing relevance and revenue in ad search: a query substitution approach. In: SIGIR (2008)
26. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML, pp. 282–289 (2001)
27. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2, 83–97 (1955)

Privacy beyond Single Sensitive Attribute

Yuan Fang[1,2], Mafruz Zaman Ashrafi[2], and See Kiong Ng[2,3]

[1] University of Illinois at Urbana-Champaign, United States
[2] Institute for Infocomm Research, Singapore
[3] Singapore University of Technology and Design, Singapore
fang2@illinois.edu, mafruz@gmail.com, skng@i2r.a-star.edu.sg

Abstract. Publishing individual specific microdata has serious privacy implications. The k-anonymity model has been proposed to prevent identity disclosure from microdata, and the work on ℓ-diversity and t-closeness attempt to address attribute disclosure. However, most current work only deal with publishing microdata with a single sensitive attribute (SA), whereas real life scenarios often involve microdata with *multiple* SAs that may be *multi-valued*. This paper explores the issue of attribute disclosure in such scenarios. We propose a method called CODIP (Complete Disjoint Projections) that outlines a general solution to deal with the shortcomings in a naïve approach. We also introduce two measures, Association Loss Ratio and Information Exposure Ratio, to quantify data quality and privacy, respectively. We further propose a heuristic CODIP* for CODIP, which obtains a good trade-off in data quality and privacy. Finally, initial experiments show that CODIP* is practically useful on varying numbers of SAs.

1 Introduction

Individual specific microdata is essential for advancing empirical research, yet publishing such data can pose serious risks to individual privacy. To minimize the privacy risks, prior methods in k-anonymity [17,16] and its variants [21,6,11], ℓ-diversity [18,13,14,23] and t-closeness [9,10] emphasized on reducing *identity disclosure* and *attribute disclosure* [7]. While these efforts help protect individual privacy to a certain degree, attribute disclosure can still occur if the microdata consist of multiple sensitive attributes (SA). We highlight two shortcomings of the prior methods leading to attribute disclosure in the presence of multiple SAs.

First, prior privacy protection methods is often insufficient when there are multiple SAs, as it is **difficult to ensure good diversity or strong closeness for every SA**.

Example 1. Consider the raw microdata in Table 1(a). Suppose race and sex are quasi-identifiers (QID) [17] and the rest are SAs. We consider all possible 2-anonymized tables as shown in Table 1(b), (c) and (d). If we only want to publish a single SA, say diagnosis, then we can publish either Table 1(c) or (d), as either table each has two distinct diagnoses in every equivalence class (which is a set of tuples that have identical values in QIDs [9]), E_1 and E_2. In addition, both tables satisfy 0.25-closeness [9]. However, if we want to publish all three SAs, each of the 2-anonymized tables has only one distinct value for one of the SAs in some equivalence class (italicized in Table 1(b), (c) and (d)). Consequently, each table only achieves 0.5-closeness for that attribute. □

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 187–201, 2011.

Table 1. Raw (a) and 2-anonymized tables (b)-(d). E_i are the equivalence classes after 2-anonymization. Abbreviations used: HT (hypertension), DB (diabetes), AS (asthma).

(a) Raw microdata

	race	sex	diagnosis	family_history	job
t_1	white	f	HT	HT	teacher
t_2	white	m	HT	DB	lawyer
t_3	white	m	DB	HT	farmer
t_4	black	m	AS	AS	teacher

(b) Anonymized. $E_1 = \{t_1, t_2\}$, $E_2 = \{t_3, t_4\}$

race	sex	diagnosis	family_history	job
white	*	*HT*	HT	teacher
white	*	*HT*	DB	lawyer
*	m	DB	HT	farmer
*	m	AS	AS	teacher

(c) Anonymized. $E_1 = \{t_1, t_3\}$, $E_2 = \{t_2, t_4\}$

race	sex	diagnosis	family_history	job
white	*	HT	*HT*	teacher
white	*	DB	*HT*	farmer
*	m	HT	DB	lawyer
*	m	AS	AS	teacher

(d) Anonymized. $E_1 = \{t_1, t_4\}$, $E_2 = \{t_2, t_3\}$

race	sex	diagnosis	family_history	job
*	*	HT	HT	*teacher*
*	*	AS	AS	*teacher*
white	m	HT	DB	lawyer
white	m	DB	HT	farmer

Second, when there are multiple SAs, a new type of attack named **background-join attack** emerges. In this new attack, we assume the adversary has some external background knowledge about some individual in the table. By joining the background knowledge and the table, s/he can deduce sensitive information.

Example 2. Suppose Table 1(b) is published. Eve links Bob to equivalence class E_2 based on his QIDs. In E_2 each SA takes two distinct values. Thus, if Eve only focuses on the SA of her interest, say diagnosis, she cannot infer whether Bob has asthma with a probability more than 0.5. However, if Eve has background knowledge that Bob is a teacher, she can deduce that Bob has asthma based on the natural join of "teacher" and the last row of the table. □

Beyond the toy example in Table 1, in real life, microdata that involve multiple SAs are also common. For instance, the dataset "Income Census (KDD)" [1] is extracted from population surveys, which involves many SAs, such as employment_status and wage_per_hour. Publishing such microdata enables useful data mining applications such as classification and association study among different SAs. However, as we have presented, prior methods have two shortcomings in dealing with multiple SAs.

Additionally, an SA can also be *multi-valued* as opposed to *mono-valued*. Given a set of values S, a mono-valued attribute can take only a value v such that $v \in S$.

However, a multi-valued attribute can take any set of values S' such that $S' \subseteq S$. Each value in S is atomic, i.e., there is no nested multi-values within a value. For instance, diagnosis is multi-valued and can take a set of values, say $\{$DB, HT, AS$\}$. In the dataset "Income Census (KDD)" [1], the attribute household_status can be regarded as multi-valued (see Sect. 7). In relational databases, multi-valued attributes are also common, although they are normalized and stored in a separate table. Since normalization is a *lossless* process, normalized tables are thus no different from the original table with multi-valued attributes from an adversary's perspective.

In this paper, we explore privacy methods for publishing microdata with multiple SAs, some of which may be multi-valued. In summary, this paper makes the following contributions:

1. We identified two drawbacks of prior methods on microdata with multiple SAs;
2. We derived a general framework CODIP to address these drawbacks, which can also be applied on multi-valued SAs;

3. We introduced two new measures *Association Loss Ratio* and *Information Exposure Ratio* that quantify data quality and privacy in the new scenario;
4. We proposed a heuristic CODIP* for CODIP, which obtains a good trade-off in data quality and privacy.

2 Related Work

Microdata are usually modelled as a table, where each row corresponds to a tuple for an individual, and each column corresponds to an attribute. It is often assumed that each tuple maps to one individual and no two tuples correspond to the same individual [20].

To prevent identity disclosure, Sweeney proposed k-anonymity [17,16], which introduced the notion of *quasi-identifiers* (QIDs). The set of tuples that have identical values in QIDs are defined as an *equivalence class* [9]. The requirement is each equivalence class must contain at least k tuples. A few variants of k-anonymity also exist, e.g., Anatomy [21] which bucketizes sensitive values instead of QIDs, Micro-aggregation [6,15] and Slicing [11]. While k-anonymity can prevent identity disclosure, it does not prevent attribute disclosure.

Recent extensions of k-anonymity also address attribute disclosure. Their philosophy is to make SA values in each equivalence class more diverse. Ref. [18] proposed p-sensitivity, requiring an SA to take at least p distinct values in every equivalence class. Furthermore, [14] pointed out that the distinct values must be "well represented", and proposed ℓ-diversity based on information entropy and attribute value frequency. In a similar spirit as ℓ-diversity, (k, e)-anonymity [23] can be adopted on continuous values such that each equivalence class must contain sensitive values of a range at least e.

However, according to [9,10], ℓ-diversity is unnecessary and difficult to achieve in some cases, and is prone to skewness and similarity attacks. To address these limitations, Li *et al.* [9] proposed t-closeness. The model requires that the distribution D_j of the SA in each equivalence class E_j is close enough to the overall distribution D in the entire table. Specifically, a table satisfies t-closeness if $\forall_{E_j} : \mathrm{dist}(D_j, D) \leq t$, where $\mathrm{dist}(X, X')$ is the Earth Mover's Distance (EMD) between X and X'. A strong closeness (i.e., a small value of closeness) indicates that the distributions of the SA in each equivalence class are similar to the overall distribution in the entire table, therefore implying less risk for attribute disclosure. Li *et al.* also introduced (n, t)-closeness [10], an extension of the basic t-closeness, which allows more flexibility while retaining closeness.

The above works only deal with a single mono-valued SA. They cannot cope with multiple SAs, with the two drawbacks identified in Sect. 1. This paper extends existing models such as k-anonymity and t-closeness to the new scenario. Currently only a limited number of works deal with multiple mono-valued SAs [12,22,4]. However, they did not deal with the background-join attack (see Sect. 1), a major problem in the presence of multiple SAs– simply because all these works publish the SAs in one table, preserving associations among SAs. Hence an adversary can join the table with his/her background knowledge to reveal other SAs. In addition, they did not address the problem of multi-valued attributes, which we will explore in this paper.

Notation	Representation
$\mathcal{A} = \{A_1, \ldots, A_s\}$	the set of sensitive attributes (SA), $s = \lvert\mathcal{A}\rvert$
$\mathcal{Q} = \{Q_1, \ldots, Q_q\}$	the set of QIDs, $q = \lvert\mathcal{Q}\rvert$
$\mathcal{E} = \{E_1, \ldots, E_c\}$	the set of equivalence classes, $c = \lvert\mathcal{E}\rvert$
D_i	the distribution of A_i in the entire table
D_{ij}	the distribution of A_i in E_j
$\mathrm{dist}(X, X')$	EMD between distributions X and X'

Fig. 1. Notations

3 CODIP: A General Solution

In this section, we first introduce a naïve t-closeness approach, which is a straightforward adaptation of t-closeness in the presence of multiple mono- or multi-valued SAs. Next, we identify the shortcomings in the naïve approach, and propose a general solution CODIP to tackle the shortcomings. Note that although our discussion is based on t-closeness, our approaches also apply to other privacy models such as ℓ-diversity and (n, t)-closeness in a similar fashion. For ease of discussion, we present a list of notations in Fig. 1.

3.1 Naïve t-Closeness Approach

Multiple mono-valued SAs. First consider only multiple mono-valued SAs. Given a k-anonymized table T, suppose all SAs $A_1, \ldots, A_s$ are mono-valued. If two SAs have strong dependency, their joint distribution would be similar to that of a single SA. In this case, we can simply consider the closeness of their distributions individually. If the SAs have weak dependency, their joint values will be very diverse, especially when the number of such SAs are large (the curse of dimensionality). In this case, it is meaningless to require an equivalence class to be "well represented" in terms of the joint values of the SAs.

As such, we define t-closeness of T based on individual SAs instead of their joint distributions. Essentially, in the naïve approach, in order for T to satisfy t-closeness, every SA must satisfy t-closeness for a given k-anonymization.

Definition 1. *A k-anonymized table T, whose SAs are all mono-valued, is said to satisfy* **t-*closeness*** *iff* $\forall_{A_i \in \mathcal{A}} \forall_{E_j \in \mathcal{E}} : \mathrm{dist}(D_{ij}, D_i) \leq t$. □

Multi-valued SAs. Given a raw table with n tuples $t_1, \ldots, t_n$, suppose there is a multi-valued SA B. B can take a subset of values in S, i.e., $\forall_{t_u} : t_u.B \subseteq S$, where $S = \{v_1, \ldots, v_m\}$. Without loss of generality, we assume the values in S are categorical, since continuous values can be discretized. It is easy to transform B into multiple mono-valued attributes.

Definition 2. *$\forall v_i \in S$, define a bit vector $(b_{i1}, b_{i2}, \ldots, b_{in})$, where each $b_{iu} = 1$ if $v_i \in t_u.B$, and $b_{iu} = 0$ otherwise. Attribute B is replaced with m mono-valued attributes $B_1, \ldots, B_m$, such that $\forall_{t_u, B_i} : t_u.B_i = b_{iu}$. We term this process* ***bitmap transformation****, and each B_i the* ***derived attribute*** *of B.* □

Informally, B is transformed into an $m \times n$ bitmap. Note that bitmap transformation is *lossless* and thus does not compromise data quality. In addition, each derived attribute is mono-valued. This allow us to adapt the naïve t-closeness approach on a bitmap transformed table as we have just discussed. We then treat the derived attributes no different from the original mono-valued attributes.

Shortcomings. The naïve approach is a direct adaptation of t-closeness, which suffers the two shortcomings in Sect. 1. We claim that the two shortcomings generally become more severe when there are more SAs.

In the context of t-closeness, the first shortcoming is that we generally have weaker closeness (i.e., a larger value of closeness) when there are more SAs, owing to the *effect of diminishing closeness*. This effect is formalized in Theorem. 1. Its proof is omitted due to space constraint.

Theorem 1. *Given a bitmap transformed table T, let T' be the projection of T on $\mathcal{A}' \cup \mathcal{Q}$, where $\mathcal{A}' \subseteq \mathcal{A}$. Let t_{best} and t'_{best} denote the best closeness that at least one k-anonymized T and T' can satisfy, respectively. The* ***effect of diminishing closeness*** *states that $t_{best} \geq t'_{best}$.*

The second shortcoming is that the threat of background-join attacks (abbreviated as "*join-threat*" hereafter) becomes greater as the number of SAs increases. When there are more SAs, an adversary can deduce new information on more SAs in a background-join attack, which increases the join-threat.

Since we are enforcing t-closeness (or other models) on each SA, the threat of traditional background attack on an individual SA is similar to that in the scenario with a single SA. We do not discuss this kind of attack as it has been addressed in previous works involving a single SA. Instead, we focus on the new threat that arises due to the existence of multiple SAs, the so-called "join-threat".

3.2 CODIP: Overcoming the Shortcomings

If we can reduce the number of SAs in a published table, we can alleviate the effect of diminishing closeness and the join-threat. Based on this, we propose a general solution called *Complete Disjoint Projections* or CODIP. In essence, CODIP projects the raw table on subsets of the SAs, and publishes the *projected tables* instead. Each projected table has a smaller number of SAs than the raw table has. Additionally, all of the SAs must be in exactly one of the projection. Formally, we call how CODIP projects the raw table a *projection plan*, or simply a *plan*.

Definition 3. *A* ***projection plan*** *projects a bitmap transformed table on its subsets of attributes $\mathcal{A}_1 \cup \mathcal{Q}, \ldots, \mathcal{A}_r \cup \mathcal{Q}$, such that (i) $\cup_{u=1}^{r} \mathcal{A}_u = \mathcal{A}$; (ii) $\forall_u : \mathcal{A}_u \neq \emptyset$; (iii) $\forall_{uw, u \neq w} : \mathcal{A}_u \cap \mathcal{A}_w = \emptyset$. We denote this plan $\Phi(\mathcal{A}_1, \ldots, \mathcal{A}_r)$. The projections are called the* ***projected tables*** *of the plan. A plan satisfies t-closeness iff every projected table satisfies t-closeness as in Def. 1.* □

To put in words, a projection plan isolates *disjoint* subsets of SAs in separate tables. Each projected table is then subjected to various anonymity algorithms, and the order of the tuples in each table is randomized. In this paper, we apply k-anonymity [17]

and t-closeness [9]; however, we stress that CODIP is a flexible framework that may adopt any previous privacy models on each projected table. The philosophy is to devise a good projection plan (see Sect. 3.3) so that any algorithm intended for a single SA (e.g., [21, 6, 13, 10]) would also work well on each projected table without suffering significantly from diminishing closeness and the join-threat, while at the same time preserving most of the utility. The SAs of each individual within any projected table is thus protected by such previous algorithms, which offers certain level of protection even in the worst case. Linking SA values across tables is also limited, as will be shown in Sect. 6.1. We will further discuss possible attacks in Sect. 6, which would not succeed on CODIP.

Clearly, the naïve t-closeness approach is a special plan (i.e., $\phi_1 = \Phi(\mathcal{A})$). On the other hand, $\phi_2 = \Phi(\{A_1\}, \ldots, \{A_s\})$ is also a special plan that publishes each SA in a separate table. In this plan, the two shortcomings are completely eliminated, since each table only contains a single SA. Note that all plans except ϕ_1 suffer some information loss. In particular, some of the associations among SA values are lost, as the correspondence of tuples from different projected tables is disturbed. Such loss of associations mitigates the shortcomings of the naïve approach at the cost of data quality. Consider ϕ_2, in which the shortcomings of ϕ_1 are completely eliminated. However, as each projected table only contains a single SA, all associations between any two SAs are lost, making data much less useful. The goal is to overcome the shortcomings as well as to minimize association loss, as we shall discuss next.

3.3 Choosing Better Plans

An optimal plan minimizes the effect of diminishing closeness, association loss, and join-threat. We have made two observations towards such an optimal plan.

Observation 1. *If two SAs have strong dependency, a background-join attack on them reveals less new information beyond the adversary's background knowledge on one of the attribute. In addition, one of them can be closely represented by the other, effectively resulting in fewer than two (independent) attributes. Thus the effect of diminishing closeness on them is less pronounced.* □

Observation 2. *If two SAs are independent or with weak dependency, their joint distribution is insignificant, as no strong associations can be inferred from it. Thus association loss is small if their joint distribution is lost.* □

We use an example to illustrate the intuitions of the two observations.

Example 3. (**Observation 1**) Suppose diagnosis and family_history are two SAs with strong dependency. If they are published in the same table, Eve (who knows Bob has hypertension), learns that Bob has a family history of hypertension by a background-join attack. However, this privacy breach is less serious, as it is quite expected given that Bob has hypertension. Moreover, since the distributions of both attributes would be similar due to their dependency, the effect of diminishing closeness is less pronounced. (**Observation 2**) Suppose job and alcohol are two independent SAs. Their joint distribution appears random– people have different drinking habits regardless of their jobs. The associations between the two provide little information beyond random guessing. Thus, we can afford to lose such associations by publishing them in different tables. □

Algorithm CODIP (T, k, t, α, β)
Input: T, a raw table containing microdata.
k, anonymity requirement.
t, closeness requirement.
α, threshold on association loss.
β, threshold on join-threat.
Output: ϕ, a projection plan.
P, the set of projected tables for ϕ.
1) Apply bitmap transformation on T;
2) Partition $\mathcal{A}$ into r disjoint subsets $\mathcal{A}_1, \ldots, \mathcal{A}_r$;
3) $\phi \leftarrow \Phi(\mathcal{A}_1, \ldots, \mathcal{A}_r)$;
4) $P \leftarrow$ CheckPlan();
5) **if** $P = null$ **then**
6) **return** *failure*;
else
7) **return** (ϕ, P);

Subroutine CheckPlan()
Input: all variables accessible in CODIP.
Output: the set of projected tables for ϕ.
8) **if** association loss in $\phi > \alpha$ **then return** *null*;
9) **if** join-threat in $\phi > \beta$ **then return** *null*;
10) **for** $i \leftarrow 1$ **to** r **do**
11) $T_i \leftarrow$ projection of T on $\mathcal{A}_i \cup \mathcal{Q}$;
12) **if** no k-anonymized T_i satisfies t-closeness **then**
13) **return** *null*;
else
14) $T_i \leftarrow$ a k-anonymized T_i satisfying t-closeness;
endfor
15) **return** $\{T_1, T_2, \ldots, T_r\}$;

Fig. 2. General framework for CODIP

To leverage the two observations, we propose a general framework for CODIP as shown in Fig. 2– a high level abstraction assuming an ideal partitioning of SAs (a concrete algorithm is proposed in Sect. 5). It requires the following user inputs: (i) T, the raw microdata table to be published; (ii) k, the anonymity requirement; (iii) t, the closeness requirement; (iv) α, the association loss threshold; (v) β, the join-threat threshold.

For inputs (iv) and (v), we delay the discussion of measuring association loss and join-threat to Sect. 4. For now, assume that they can be quantified. Also, assume that users can specify appropriate values for α and β, following the discussion on their relationships in the experiments (Sect. 7.1), although a more extensive study on this issue is beyond the scope of this paper.

The key operation lies in Step 2, which partitions $\mathcal{A}$ into disjoint subsets. Ideally, the partitioning should be consistent with Observation 1 and 2. In reality it only needs to be consistent to such a degree that a "sufficiently good" plan is obtained, which satisfies user specified thresholds t, α and β. Step 4 invokes the subroutine CheckPlan(), which examines if the plan satisfies the thresholds. If so, it returns a set of k-anonymized projected tables; otherwise, it fails. Note that users can optionally impose a quality threshold on QIDs in k-anonymization (Step 12 and 14), e.g., discernibility metric [2].

In this general framework, we do not enforce any specific algorithm to achieve a "sufficiently good" partitioning. A brute force method that enumerates all possible ways of partitioning and then selects one is infeasible since the number of possible ways to partition a set is intractable. We will propose an efficient heuristic CODIP* in Sect. 5 without requiring the costly enumeration.

4 Evaluating Projection Plans

In addition to k-anonymity that measures anonymity and t-closeness that measures closeness, we propose two more measures on a projection plan for CODIP: (1) *Association Loss Ratio* (Γ_α), the degree of association loss due to the lost joint distributions of the SAs; and (2) *Information Exposure Ratio* (Γ_β), the level of join-threat due to background-join attacks. The measures are based on *mutual information* (MI) [5], which can quantify nonlinear dependency between attributes, as opposed to correlation which only measures linear relationships. It means MI can detect dependency caused

by not only positive or negative correlations, but also "mixed" correlations. Hence, it is well-suited for formally capturing the notion of dependency in Observations 1 and 2.

4.1 Association Loss Ratio

We propose a measure to quantify association when SAs are projected onto different tables. Given a bitmap transformed table, for a pair of SAs A_i and A_j, their MI is $I(A_i, A_j) = \sum_{v \in A_i, v' \in A_j} p(v, v') \log\left(\frac{p(v,v')}{p(v)p(v')}\right)$ [5], where $p(x)$ is the pmf of attribute X, and $p(x, y)$ is the joint pmf of X and Y.[1] MI quantifies how much information two attributes share, which also implies the degree of independence between them. In particular $I(A_i, A_j) = 0$ if A_i and A_j are independent. We can use it to quantify how significant the association between the values of A_i and A_j are (Observation 2). Lower MI suggests a higher degree of independence, and thus the association between their values is less significant. This further implies that association loss is smaller if the joint distribution of the two attributes becomes unknown.

By computing the fraction of MI of all pairwise SAs whose joint distributions are unknown, we obtain a [0,1]-normalized measure of association loss– Association Loss Ratio (Γ_α). Given a projection plan $\phi = \Phi(\mathcal{A}_1, \ldots, \mathcal{A}_r)$ such that $\cup_{u=1}^{r} \mathcal{A}_u = \mathcal{A} = \{A_1, \ldots, A_s\}$, the sum of all pairwise MI is $I_\Sigma(\phi) = \frac{1}{2}\sum_{i,j \neq i} I(A_i, A_j)$, and the sum of unknown pairwise MI is $I_\alpha(\phi) = \frac{1}{2}\sum_{i,j \neq i} W_\alpha(A_i, A_j) I(A_i, A_j)$, where $W_\alpha(A_i, A_j)$ assigns a boolean weight— 1 if A_i, A_j are in different projected tables, 0 otherwise (i.e., $I(A_i, A_j)$ is summed in $I_\alpha(\phi)$ only if A_i, A_j are not projected onto the same table). Association Loss Ratio is then defined as a fraction in terms of $I_\Sigma(\phi)$ and $I_\alpha(\phi)$:

$$\Gamma_\alpha(\phi) = \begin{cases} I_\alpha(\phi)/I_\Sigma(\phi) & \text{if } I_\Sigma(\phi) \neq 0; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Note the special cases that $\Gamma_\alpha(\Phi(\mathcal{A})) = 0$, and $\Gamma_\alpha(\Phi(\{A_1\}, \ldots, \{A_s\})) = 1$. In general, $\Gamma_\alpha(\phi)$ is smaller if the plan ϕ is generated in compliance with Observation 2.

4.2 Information Exposure Ratio

Next, we propose a measure of information exposure resulted from background-join attacks to indicate the level of join-threat. Clearly, when more *new* information is exposed, the threat level is higher. Thus, any background knowledge that is already known to the adversary must be excluded.

Consider any pair of SAs A_i and A_j. Suppose their joint distribution is known to an adversary, i.e., they are published in the same projected table. Assuming that the adversary identifies a tuple and has background knowledge in one of them (say A_i), s/he can then learn the value of the other SA (A_j). Potential new information of the other SA (A_j) could be exposed to the adversary. The other two cases are trivial: (i) if the adversary knows neither A_i nor A_j, no background-join attack can be launched on them; (ii) if the adversary knows both, no new information will be exposed.

The amount of information expressed by an attribute A_i can be represented by its information theoretic entropy [5], which is defined as $H(A_i) = -\sum_{v \in A_i} p(v) \log(p(v))$.

[1] We avoid the notations $p_X(x)$ and $p_{X,Y}(x, y)$ for convenience if no ambiguity arises.

The relationship of the entropies of A_i and A_j is illustrated by the Venn diagram in Fig. 3. For any pair of SAs, if the adversary deduces the value of A_i (or A_j) based on his or her background knowledge of A_j (or A_i), the amount of new information exposed is h_1 (or h_2). Therefore, total amount of new information that can be exposed from this pair is $h_1 + h_2$. Since a larger $h_3 = I(A_i, A_j)$ results in a smaller $h_1 + h_2$, less information can be exposed to an adversary when A_i and A_j have more dependency.

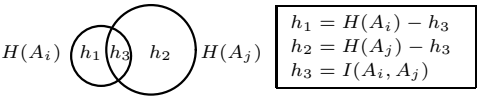


Fig. 3. Relationship of A_i and A_j's entropies

Also, some values in a SA could be non-sensitive depending on the user (e.g., nil value). Hence users should be allowed to define what constitute sensitive values in a SA. Let $\mathrm{Sens}(x)$ denotes the predicate that asserts x is a sensitive value. By default, $\mathrm{Sens}(x)$ is true for all values in a SA; however, users have the flexibility to customize it.

Subsequently we derive $E(A_i, A_j)$, the total amount of new sensitive information that is *exposable* in a pair of SAs A_i and A_j. Taking the sensitivity of values into account, it is computed by summing up exposable information for each joint value, weighted by the joint probability:

$$E(A_i, A_j) = \sum_{v \in A_i, v' \in A_j} p(v, v') \times \begin{cases} 0 & \neg\,\mathrm{Sens}(v) \wedge \neg\,\mathrm{Sens}(v'); \\ H(A_i) - I(A_i, A_j) & \mathrm{Sens}(v) \wedge \neg\,\mathrm{Sens}(v'); \\ H(A_j) - I(A_i, A_j) & \neg\,\mathrm{Sens}(v) \wedge \mathrm{Sens}(v'); \\ H(A_j) + H(A_j) - 2I(A_i, A_j) & \mathrm{Sens}(v) \wedge \mathrm{Sens}(v'). \end{cases}$$

Since a background-join attack is confined within the projected tables that contain the attributes on which the adversary has background knowledge, we compute the fraction of exposable information for each projected table. Given a projection plan $\phi = \Phi(\mathcal{A}_1, \ldots, \mathcal{A}_r)$ such that $\cup_{u=1}^{r} \mathcal{A}_u = \mathcal{A} = \{A_1, \ldots, A_s\}$, the sum of exposable information in all pairwise SAs (i.e., assuming there is only one projected table) is $E_\Sigma(\phi) = \frac{1}{2} \sum_{i,j \neq i} E(A_i, A_j)$, and the sum of *actual exposed* information in all pairwise SAs in the projected table on $\mathcal{A}_u \cup \mathcal{Q}$ can be computed as $E_\beta(\mathcal{A}_u) = \frac{1}{2} \sum_{i,j \neq i} W_\beta(A_i, A_j, \mathcal{A}_u) E(A_i, A_j)$, where $W_\beta(A_i, A_j, \mathcal{A}_u)$ assigns a boolean weight—1 if $A_i \in \mathcal{A}_u$ and $A_j \in \mathcal{A}_u$, and 0 otherwise (i.e., only actual exposed information in the projected table on $\mathcal{A}_u \cup \mathcal{Q}$ is summed). Information Exposure Ratio (Γ_β) is then defined as the sum of fractions in terms of $E_\beta(\mathcal{A}_u)$ and $E_\Sigma(\phi)$ for each projected table, normalized by the number of SAs in that table:

$$\Gamma_\beta(\phi) = \begin{cases} \sum_{u=1}^{r} \left(\frac{E_\beta(\mathcal{A}_u)}{E_\Sigma(\phi)} \cdot \frac{|\mathcal{A}_u|}{|\mathcal{A}|} \right) & \text{if } E_\Sigma(\phi) \neq 0; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Note the special cases that $\Gamma_\beta(\Phi(\{A_1\}, \ldots, \{A_s\})) = 0$, and $\Gamma_\beta(\Phi(\mathcal{A})) = 1$. In general, $\Gamma_\beta(\phi)$ is smaller if the plan ϕ is generated in compliance with Observation 1.

Algorithm CODIP* (T, k, t, α, β)
Input/Output: same as CODIP.
1) Apply bitmap transformation on T;
2) **for** $i = 1$ **to** s **do** $\mathcal{A}_i \leftarrow \{A_i\}$;
3) $\phi \leftarrow \Phi(\mathcal{A}_1, \ldots, \mathcal{A}_s)$;
4) $P \leftarrow$ CheckPlan*();
5) **if** $P = null$ **then return** *failure*;
6) **repeat**
7) $(\mathcal{A}_u, \mathcal{A}_w) \leftarrow \text{argmax}_{u,w:u\neq w} \text{AvgI}(\mathcal{A}_u \cup \mathcal{A}_w)$;
8) $\phi' \leftarrow \phi$; /* temp. placeholder */

(Continued)
9) $P' \leftarrow P$; /* temp. placeholder */
10) Remove $\mathcal{A}_u, \mathcal{A}_w$ from ϕ;
11) Add $\mathcal{A}_u \cup \mathcal{A}_w$ to ϕ;
12) $P \leftarrow$ CheckPlan*();
13) **until** $P = null$;
14) **if** $\Gamma_\alpha(\phi') \leq \alpha$ **then**
15) **return** (ϕ', P');
else
16) **return** *failure*;

Fig. 4. Outline of CODIP*

4.3 Evaluation of Plans

We use Association Loss Ratio and Information Exposure Ratio to evaluate the quality of a projection plan for CODIP. Based on their definitions, smaller ratios indicate a better plan. We propose to evaluate our plans against a baseline, the naïve t-closeness approach in Sect. 3.1, i.e., the plan $\Phi(\mathcal{A})$. By Theorem 1, $\Phi(\mathcal{A})$ has the weakest closeness among all plans. Furthermore, $\Gamma_\alpha(\Phi(\mathcal{A})) = 0$ and $\Gamma_\beta(\Phi(\mathcal{A})) = 1$. Given a plan ϕ, suppose $\Gamma_\alpha(\phi) = \alpha$, $\Gamma_\beta(\phi) = \beta$, ϕ satisfies t'-closeness and $\Phi(\mathcal{A})$ satisfies t-closeness. We say ϕ has a $(1 - t'/t) \times 100\%$ improvement in closeness, $(1 - \beta) \times 100\%$ reduced join-threat, while suffers $\alpha \times 100\%$ association loss, as compared to the naïve t-closeness approach.

5 CODIP*: A Heuristic for CODIP

In the CODIP framework proposed in Sect. 3.2, we have not described a suitable algorithm for generating good plans. A brute force approach to enumerate all possible plans is infeasible on high dimensional data. Thereby we propose a bottom-up greedy heuristic CODIP*, outlined in Fig. 4.

We start bottom-up from the initial plan $\phi = \Phi(\{A_1\}, \ldots, \{A_s\})$ (Steps 2–3). The basic idea is to ignore $\Gamma_\alpha(\phi)$ first, and merge the disjoint subsets of SAs in ϕ as much as possible. In this way, we attempt to reduce $\Gamma_\alpha(\phi)$ below its threshold while avoid exceeding closeness and $\Gamma_\beta(\phi)$ thresholds. The key operations lie in Steps 6–13, which correspond to the partitioning operation in CODIP (Step 2 in Fig. 2). We greedily pick two subsets of SAs $\mathcal{A}_u$ and $\mathcal{A}_w$ from the plan ϕ, such that the average pairwise MI in $\mathcal{A}_u \cup \mathcal{A}_w$ (AvgI in Step 7) is maximized. We then merge the two subsets $\mathcal{A}_u$ and $\mathcal{A}_w$ in ϕ (Steps 10 and 11). Based on Observations 1 and 2, this merging would greatly reduce $\Gamma_\alpha(\phi)$, and result in a small increase in closeness and $\Gamma_\beta(\phi)$ at least locally. The merging process is repeated until the plan ϕ exceeds the thresholds on closeness or $\Gamma_\beta(\phi)$ (Step 6-13). The subroutine CheckPlan*() checks if ϕ satisfies the thresholds on closeness and Γ_β. It is identical to CheckPlan() in CODIP, except that it does not check for Γ_α (i.e., eliminate Step 8 in Fig. 2), as it will be checked later. Subsequently, the plan before the last merger is returned if it satisfies the threshold on Γ_α (Step 14–16).

CODIP* is efficient by avoiding the combinatorial enumeration of attributes. For a dataset with s number of SAs, in the worst case, only $s - 1$ mergers are necessary (i.e., the number of repetitions of Step 6–13 is bounded by $O(s)$).

6 Discussion of Possible Attacks on CODIP

6.1 Intersection Attack

Intersection attack occurs when multiple tables are intersected on common attributes [19], potentially re-establishing the links among sensitive values and QIDs across tables. [19] proposed the notion of (X, Y)-linkability– the extent of "linking" between X (QIDs) and Y (SAs). (X, Y)-linkability is satisfied if the confidence of inferring any value on Y from any value on X (can be joint value on X or Y) does not exceed a threshold $\epsilon \in (0, 1]$. We show that releasing multiple tables using CODIP introduces no more linking risk than releasing a single table using k-anonymity and distinct-ℓ-diversity, i.e., each equivalence class must contain at least ℓ distinct values, $\ell \geq 2$.

Theorem 2. *The tables released by CODIP (each table protected by k-anonymity and distinct-ℓ-diversity) satisfies (X, Y)-linkability with a threshold the same as the case of a single table released using k-anonymity and distinct-ℓ-diversity.*

Proof. As the subset of SAs (Y) in each projected table is disjoint, only QIDs (X) can be intersected. Consider a join of m tables by intersecting on some QIDs. By k-anonymity there are at least k tuples in each table with the same QIDs, producing a join with at least k^m tuples for any (joint) value on QIDs. Among the k^m or more joint tuples, we examine how many have the same value on some SAs. By ℓ-diversity there is at most $k - \ell + 1$ instances for any (joint) value on any SAs from one table. This follows that there are at most $k^{m-p}(k - \ell + 1)^p$ instances for any (joint) value on SAs from p tables ($1 \leq p \leq m$). Thus the confidence of infering SAs from QIDs is at most $\frac{k^{m-p}(k-\ell+1)^p}{k^m} = (\frac{k-\ell+1}{k})^p \leq \frac{k-\ell+1}{k}$. The upperbound is the threshold, which is independent of m. That means the same threshold is obtained when $m = 1$, i.e., a single table using k-anonymity and distinct-ℓ-diversity is released. □

Another type of intersection attack is targeted at incremental releases [3], where new tuples for the same schema are included and re-released with old tuples. Sensitive values can be intersected among old and new releases to derive hidden information. This type of attack is inapplicable to CODIP for two reasons: (i) in each projected table, the tuples all refer to the same set of individuals (i.e., no old and new tuples); (ii) given that there are no common SAs across tables, intersection on sensitive values is not possible.

6.2 Minimality Attack

Minimality attack [20], is possible if the adversary knows the privacy algorithm. The attack utilizes the concept of "minimality", as most privacy algorithms attempt to minimize information loss in order to preserve utility.

For CODIP, minimality attack is possible on two levels. *First*, minimality attack can target at each projected table, where k-anonymity is enforced. In this case, the m-confidentiality model [20] can be applied on each projected table to counter minimality attacks. *Second*, minimality attack can potentially target to restore the correspondence of tuples in different tables. Fortunately, CODIP is not vulnerable to this. While CODIP attempts to minimize the Information Loss Ratio Γ_α, its notion of minimization is relative to the MI of all pairwise attributes, and not to all possible correspondence of tuples

from different tables. Even if an adversary has obtained a correspondence of tuples with smallest possible Γ_α, this smallest Γ_α does not indicate a correct correspondence of tuples.

7 Experiments

We performed some initial experiments to study the trade-off between data quality and privacy. We choose the naïve t-closeness approach in Sect. 3.1 as our baseline. Note that the approaches in [12, 22, 4] publish all SAs in one table, thus they are vulnerable to background-join attacks in the exact same way as the baseline. Therefore it is fair to compare CODIP* with the baseline only, which suffers the same problem as these previous work. Moreover, to achieve k-anonymity, we adopted a full-domain generalization scheme as outlined in Incognito [8].

The "Census-Income (KDD)" training dataset [1] is used. We chose four QIDs– age, race, sex, citizenship, as well as SAs – seven categorical (worker_class, education, industry, employment_status, business_status, salary_class, occupation), four numeric discretized to $\{0, 1\}$ (wage_per_hour, dividend, capital_gain, capital_loss), and one multi-valued (household_status, giving four derived attributes married, 18^-, descendent, sub-family). There are effectively a total of 15 SAs. Additionally, tuples with missing or unknown values are discarded, giving a total of 98839 tuples that remain.

All algorithms were implemented in Java. The experiments were conducted on a 3.0GHz PC with 3GB memory.

7.1 Relationship of Γ_α and Γ_β

Intuitively, given a plan ϕ, a larger $\Gamma_\alpha(\phi)$ implies a smaller $\Gamma_\beta(\phi)$. This experiment studies the relationship between Association Loss Ratio and Information Exposure Ratio. Since the two ratios only depend on the way the raw table is projected, k-anonymity and t-closeness requirements does not affect them.

We run CODIP* with varying thresholds. Starting from $\beta = 1$, which is the threshold on $\Gamma_\beta(\phi)$, we gradually decrease it. For each β value, we record the smallest $\Gamma_\alpha(\phi)$ that has incurred. A plot of $\Gamma_\beta(\phi)$ against $\Gamma_\alpha(\phi)$ is presented in Fig. 5, where ϕ is the plan generated by CODIP* given a threshold β.

In Fig. 5, when no association loss incurs, i.e., $\Gamma_\alpha(\phi) = 0$, the join-threat is maximum at $\Gamma_\beta(\phi) = 1$. However, if we slightly relax $\Gamma_\alpha(\phi)$, we can trade for a significant reduction in $\Gamma_\beta(\phi)$. This is evident from a sharp decrease in $\Gamma_\beta(\phi)$ from 1 to 0.15, when $\Gamma_\alpha(\phi)$ slowly increases from 0 to 0.19. However, to further reduce the join-threat, a small decrease in $\Gamma_\beta(\phi)$ would result in a drastic increase in $\Gamma_\alpha(\phi)$, which is a less desirable trade-off. Generally, we can get a good trade-off plan if we allow some association loss and join-threat, without attempting to eliminate either factor or impose an extremely small threshold.

Next, we study the effects of the number of projected tables (N) on the plans. We evaluate the plans generated by CODIP* against our baseline, the naïve t-closeness approach (i.e., $N = 1$). Fig. 6 shows the results of our experiment.

As expected, when there are fewer projected tables in a plan, privacy is less protected as shown by the lesser reduction in join-threat in Fig. 6. On the other hand, data quality

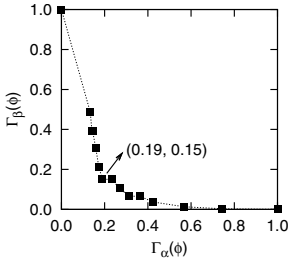


Fig. 5. Relationship of $\Gamma_\beta(\phi)$ and $\Gamma_\alpha(\phi)$

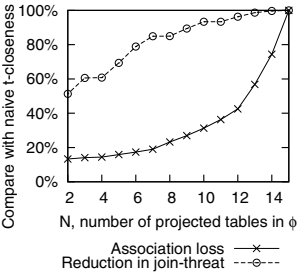


Fig. 6. Effects of no. of tables

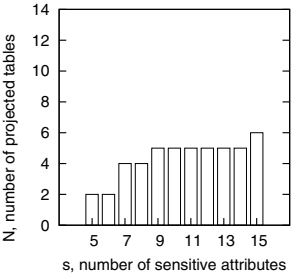


Fig. 7. Effects of no. of SAs ($\alpha = \beta = 0.3$)

improves as reflected in the decreasing association loss. This observation is consistent with CODIP*. In CODIP*, every merging action results in one fewer table causing less association loss while risking more join-threat. Note that as the number of projected tables increases, reduction in join-threat increases in a decreasing rate, whereas association loss increases in an increasing rate. Therefore, a good trade-off plan usually has a smaller number of projected tables (e.g., less than 7 in this experiment), and there are some association loss and join-threat that must be allowed (as we have just discussed based on Fig. 5).

Lastly, to show the scalability of CODIP*, we vary the number of SAs (s). The number of projected tables (N) outputted by CODIP* is shown in Fig. 7. As s increases, N also increases. However, the growth of N is minimal when s is large ($s \geq 9$ in this experiment). This result indicates that CODIP* is effective in protecting privacy while producing a small number of projected tables, even if there are a large number of SAs.

The experiments verified the possibility of greatly enhancing privacy while slightly sacrificing data quality, i.e., a good trade-off can be obtained in practice.

7.2 Closeness and Anonymity

Next, we study the closeness and anonymity requirements t and k, respectively. First, consider $k = 2$. To ensure the quality of QIDs, we also impose a discernibility metric [2] (d_m, in unit of 10^9) threshold on QIDs, such that k-anonymized tables with discernibility metric larger than d_m are not considered. Smaller d_m implies higher quality in QIDs, causing fewer number of valid anonymizations.

Following the analysis in Sect. 7.1, we set thresholds $\alpha = 0.2, \beta = 0.5$. Starting from $\beta = 0.5$, we gradually decrease it, and obtain 6 plans by CODIP*, each with a varying number of projected tables ($N \in [2, 7]$). Fig. 8 shows the best closeness achieved by the plans under different thresholds d_m for $N \in \{2, 4, 6\}$, in addition to the baseline ($N = 1$), and the special plan with each SA published in a separate table ($N = 15$). Specifically, Fig. 8(a) depicts the absolute closeness each plan can achieve at best, whereas Fig. 8(b) compares the plans with the baseline and presents the improvement of each plan.

We observe that smaller N results in weaker closeness. In CODIP*, N becomes smaller when more mergers take place, resulting in a non-decreasing number of SAs in

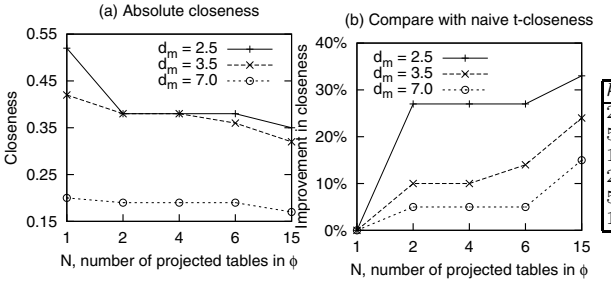


Fig. 8. Best closeness achieved by plans

(a) # of plans

k	$t = 0.4$	$t = 0.37$
2	6	2
5	6	2
10	6	2
20	6	2
50	0	0
100	0	0

(b) # of k-anonymizations

k	$t = 1$
2	46
5	28
10	18
20	14
50	5
100	1

Fig. 9. Effects of k ($\alpha = 0.2, \beta = 0.5, d_m = 3.5$)

each projected table. This result demonstrates the effect of diminishing closeness. Also note that when d_m is smaller, there are fewer valid anonymizations, resulting in weaker closeness. Hence, the improvement in closeness w.r.t. to the naïve t-closeness approach is potentially more significant.

Finally, we study the effects of k on closeness. We count the number of plans that can satisfy the various thresholds in Fig. 9. Fig. 9(a) presents our findings. Note that the baseline can only satisfy 0.42-closeness when $k \in \{2, 5, 10, 20\}$, and 0.52-closeness otherwise. When $k \leq 20$, we have quite a number of plans that can satisfy the requirements on closeness. As expected, a stronger closeness (i.e., a smaller t) results in fewer valid plans. However, when k becomes large ($k \geq 50$), there is apparently no plan that can satisfy the thresholds. The reason is that the number of valid k-anonymizations drops as k increases. Fig. 9(b) shows the number of valid k-anonymizations, assuming no requirement on closeness (i.e., $t = 1$). When k increases from 2 initially, the number of valid plans remains unaffected, as the k-anonymizations that are eliminated due to increased k are expected to have weaker closeness– the eliminated anonymizations contain at least an equivalence class whose cardinality is smaller than k, and smaller equivalence classes are generally less "well represented." When k continues to increase beyond 20, the number of valid k-anonymizations becomes too few. It is likely that none of these few satisfies the given closeness, which is indeed the case in this experiment. Results showed that if k is not too large (e.g., $k < 50$), CODIP* generates plans that satisfy stronger closeness, as compared to the baseline.

8 Conclusion

We studied the privacy issue of attribute disclosure in publishing microdata that have multiple SAs, of some may be multi-valued. We introduced Association Loss Ratio and Information Exposure Ratio to quantify data quality and privacy, respectively. We showed that a direct adaptation of t-closeness is inadequate, and proposed a framework CODIP and a heuristic CODIP*. Experiments showed that CODIP* generates good trade-off plans on a real dataset.

References

1. Asuncion, A., Newman, D.: UCI machine learning repository. Univ. of California, Irvine, ICS (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Bayardo, R., Agrawal, R.: Data privacy through optimal k-anonymization. In: ICDE, pp. 217–228 (2005)
3. Byun, J., Sohn, Y., Bertino, E., Li, N.: Secure anonymization for incremental datasets. In: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 48–63. Springer, Heidelberg (2006)
4. Chen, Z., Gangopadhyay, A.: A Privacy Protection Model for Patient Data With Multiple Sensitive Attributes. IJISP 2(3), 28–44 (2008)
5. Cover, T., Thomas, J.: Elements of information theory. Wiley, Chichester (1991)
6. Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogeneous k-anonymity through microaggregation. DMKD 11(2), 195–212 (2005)
7. Lambert, D.: Measures of disclosure risk and harm. JOS 9, 313–331 (1993)
8. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: SIGMOD, p. 60 (2005)
9. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and ℓ-diversity. In: ICDE, pp. 106–115 (2007)
10. Li, N., Li, T., Venkatasubramanian, S.: Closeness: A New Privacy Measure for Data Publishing. TKDE (June 2009)
11. Li, T., Li, N., Zhang, J., Molloy, I.: Slicing: a new approach for privacy preserving data publishing. cs.DB, arXiv preprint: 0909.2290v1
12. Li, Z., Ye, X.: Privacy protection on multiple sensitive attributes. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 141–152. Springer, Heidelberg (2007)
13. Machanavajjhala, A., Gehrke, J., Kifer, D.: ℓ-diversity: Privacy beyond k-anonymity. In: ICDE, pp. 24–35 (2006)
14. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity. TKDD 1(1), 3 (2007)
15. Solanas, A., Sebé, F., Domingo-Ferrer, J.: Micro-aggregation-based heuristics for p-sensitive k-anonymity: one step beyond. In: PAIS, pp. 61–69 (2008)
16. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. IJUFKS 10(5), 571–588 (2002)
17. Sweeney, L.: k-anonymity: A model for protecting privacy. IJUFKS 10(5), 557–570 (2002)
18. Truta, T., Vinay, B.: Privacy protection: p-sensitive k-anonymity property. In: ICDE PDM Workshop, p. 94 (2006)
19. Wang, K., Fung, B.: Anonymizing sequential releases. In: SIGKDD, p. 423 (2006)
20. Wong, R., Fu, A., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. In: VLDB, pp. 543–554 (2007)
21. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: VLDB, p. 150 (2006)
22. Ye, Y., Liu, Y., Wang, C., Lv, D., Feng, J.: Decomposition: Privacy preservation for multiple sensitive attributes. In: Zhou, X., Yokota, H., Deng, K., Liu, Q. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 486–490. Springer, Heidelberg (2009)
23. Zhang, Q., Koudas, N., Srivastava, D., Yu, T.: Aggregate query answering on anonymized tables. In: ICDE, pp. 116–125 (2007)

Privacy-Aware DaaS Services Composition

Salah-Eddine Tbahriti[1], Michael Mrissa[1], Brahim Medjahed[2], Chirine Ghedira[1], Mahmoud Barhamgi[1], and Jocelyne Fayn[3]

[1] Claude Bernard Lyon1 University, 69622, Villeurbanne, France
firstname.lastname@liris.cnrs.fr
[2] Department of Computer and Information Science, University of Michigan-Dearborn, 4901 Evergreen Road, Dearborn, MI 48128, USA
brahim@umd.umich.edu
[3] EA 4171: MTIC, Hopital Cardiologique de Lyon, Bat B13, 28 av. Doyen Lepine - 69677 BRON cedex, France
jocelyne.fayn@insa-lyon.fr

Abstract. Data as a Service (DaaS) builds on service-oriented technologies to enable fast access to data resources on the Web. However, this paradigm raises several new privacy concerns that traditional privacy models do not handle since they only focus on the service interface without taking into account privacy constraints related to the data exchanged with a DaaS during its invocation. In addition, DaaSs compositions may reveal also privacy-sensitive information. In this paper we propose a privacy formal model in order to extend DaaS descriptions with privacy capabilities. The privacy model allows a service to define a *privacy policy* and a set of *privacy requirements*. We propose also a privacy-preserving DaaS composition approach allowing to verify the compatibility between privacy requirements and policies in DaaS composition. We validate the applicability of our proposal with some experiments.

Keywords: Privacy, DaaS services, Composition, Dependency.

1 Introduction

Recent years have witnessed a growing interest in using Web services as a reliable medium for data publishing and sharing. This new type of services is known as *Data-as-a-Service* services [4] [17], corresponds generally to calls over data sources. While individual DaaS services may provide interesting information alone, in real scenarios like epidemiological studies, users' queries require the invocation of several services. The DaaS composition is a powerful solution for building value-added services on top of existing ones [15] [20]. In the context of our project PAIRSE[1] we proposed in [2] a mediator-based approach to compose DaaSs. In that approach the proposed mediator answers users complex queries by combining available DaaSs and carries out all the interactions between the

[1] This research project is supported by the French National Research Agency under grant number ANR-09-SEGI-008.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 202–216, 2011.

composed services. Depending on available DaaSs, the mediator may return a set of DaaS compositions all answering the same query. However, DaaS compositions in that approach may reveal privacy-sensitive information. Privacy preservation is indeed still one of the most challenging problems in DaaS composition. In this paper we address the privacy issue in DaaS composition. We propose a privacy formal model in order to extend DaaS descriptions with privacy capabilities. The privacy model, goes beyond traditional data-oriented models, by allowing a service to define a *privacy policy* (specifying how it treats its collected data) and a set of *privacy requirements* (specifying how it expects consumer services to treat its provided data) by defining a set of privacy rules. We propose also an annotation mechanism to link DaaSs to their defined privacy polices and requirements.

Component DaaSs in a composition may have different privacy concerns, thus leading to an incompatibility problem between the privacy policies and requirements of interconnected services. The second contribution is a privacy-aware DaaS Composition. We devise a compatibility matching algorithm to check the privacy compatibility among privacy requirements and policies within a composition. The compatibility matching is based on the notion of privacy subsumption and a cost model. A matching threshold is set up by a given service to cater for partial and total privacy compatibility.

Our paper is structured as follows. First, we overview related work in Section 2. We then describe our privacy model in Section 3. Then, we introduce the notion of compatibility between privacy policies and requirements in Section 4, and illustrate its importance in the context of DaaS composition and will show how our DaaS composition approach is extended within privacy-preserving in Section 5. We present our experiments in Section 6 and discuss future work in Section 7.

2 Related Work

Our work is inspired and informed by a number of areas. We briefly review the closely related areas below and discuss how our work leverages and advances the current state-of-the-art techniques.

2.1 Privacy Aware-Data Modeling

A typical example of modeling privacy is P3P [19] standard. It encodes privacy policies in XML for Web sites and specifies the mechanisms to locate and transport privacy policies. However, the major focus of P3P is to enable only Web sites to convey their privacy policies. The work in [18] aims at specifying DAML-S ontology to answer two questions: how sensitive the information is; and under what conditions the information has that sensitive degree. Regarding that, data providers specify how to use the service (mandatory and optional data for querying the service), while individuals specify the type of access for each part of their personal data contained in the service. However, privacy preferences do not include the point of view of individuals over the data usage. An

approach on the feasibility of achieving a balance between consumers privacy and provider search has been proposed in [21]. It allows client to collect, summarize, and organize their personal information into a hierarchical profile. Through this profile, the client controls which portion of its private information is exposed to the provider by adjusting a threshold. The work in [16] aims at protecting the content of client queries and the retrieved documents. It proposes a schema for a provider to perform similarity-based text retrieval while protecting clients search activities. In our work, privacy resource is specified and may be related to client, Data and Service providers levels, and not only to the provided data.

2.2 Privacy Aware-Composition

The works in services composition are closely inspired from workflow and Data mashups composition. In [7] a framework for enforcing data privacy in workflows is described. In [8], the use of private data is reasoned for workflows. Privacy-preserving mechanism for data mashup is represented in [13]. It aims at integrating private data from different data providers in secure manner. The authors in [12] discuss the integration and verification of privacy policies in SOA-based workflows. The previous approaches, related to data mashup and workflows, focus on using algorithms (such as k-anonymity) for preserving privacy of data in a given table, while in our work we go further and propose a model that also takes into account usage restrictions and client requirements. The works [9] [10] [6] propose using third parties as database service providers without the need for expensive cryptographic operations. However the proposed schemes do not allow queries to execute over the data of multiple providers and do not take into account the privacy issue regarding service provider and data consumer, which is the main focus of our work. In the filed of data integration, several efforts have been made to either preserve the privacy of individuals using sanitized techniques [1] [3] or to preserve the privacy of the datasource while running data integration algorithms over multiple databases using cryptographic techniques [5]such as *secure multi-party computation* and *encryption*. In contrast to the existing approaches, in this paper we introduce a service-oriented privacy model for DaaS that goes beyond "traditional" data-oriented privacy approaches. *Input/output* data as well as *operation* invocation may reveal sensitive information about services and hence, should be subject to privacy constraints.

3 Privacy Description Model

In this section, we propose a formal model to specify the privacy capabilities attached to DaaS service (simply service) description. With this model, a service $\mathtt{S}$ will define a *privacy policy* (noted as $\mathtt{PP}^{\mathtt{S}}$) specifying the set of privacy practices applicable on any collected data and *privacy requirements* (noted as $\mathtt{PR}^{\mathtt{S/T}}$) specifying the $\mathtt{S}$'s set of privacy conditions that a third-party service $\mathtt{T}$ must meet to consume its data. Indeed, privacy is a very subjective notion, for instance, a given service may consider an input parameter provided to a third-party service

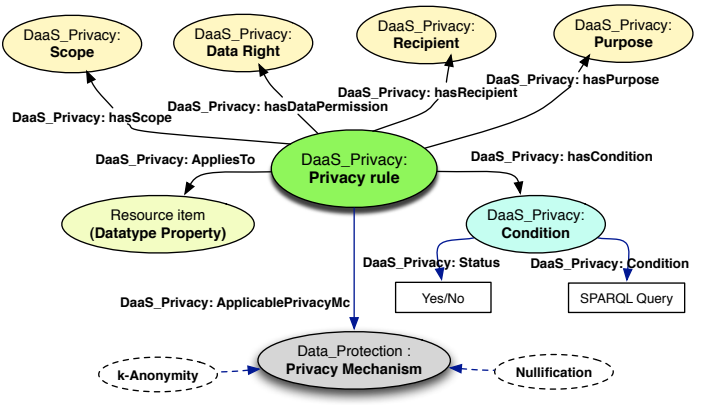


Fig. 1. Graph-based representation of a privacy rule

as private; another may view the information stating that the service invoked a specific operation of a given third-party service as private. Our model relies on the definition of privacy resource and privacy rule. Different types of information may be subject to privacy. We refer to such information as privacy resources (simply resources). To take into account the type of resources, we introduce the notion of privacy level (simply level). A graph-based representation of our model is presented in Figure 1.

3.1 Privacy Level

We define two privacy levels: data and operation. The data level deals with the data privacy. The resources (i.e., *Resource item* in Figure 1.) refer to input and output parameters of a service (e.g., defined in WSDL). For instance, service S_a has an operation op_a called *Patent-research* that takes as input a *user query* and returns as output *PatentResults*. The *user query* and *PatentResults* (i.e., input and output, resp.) may be both viewed as private; they are hence defined as data resources. The operation level copes with the privacy about operation's invocation. Information about operation invocation may be perceived as private independently on whether their input/output parameters are confidential or not [11]. For instance, let us consider a scientist that has found an invention about the causes of some infectious diseases, he invokes op_a to search if such an invention is new before he files for a patent. When conducting the query, the scientist may want to keep the invocation of op_a, query and result of query (i.e., the op_a input, op_a output resp.) private, perhaps to avoid part of his idea being stolen by a competing company. We give below the definition of the privacy level.

Definition 1. Let *rs* be a privacy resource of a service `S`. The privacy level L of *rs* is defined as follows: (i)L="data" if *rs* is an input/output of `S` operation; (ii)L="operation" if *rs* is information about `S`'s operation. ◊

3.2 Privacy Rule

The sensitivity of a resource may be defined according to several dimensions called *privacy rules*. We call the set of privacy rules *Rules Set*(RS). We define a privacy rule by a *topic*, *level*, *domain*, and *scope*. The *topic* gives the privacy facet represented by the rule. For instance, given the representation of privacy rule in Figure 1, the topic may include: the data right, the recipient and the purpose. The "purpose" topic states the intent for which a resource collected by a service will be used; the "recipient" topic specifies to whom the collected resource can be revealed. The *level* represents the privacy level on which the rule is applicable. The domain of a rule depends on its level. Indeed, each rule has one single level: "data" or "operation". We use the terms data and operation rule to refer to a rule with a "data" and "operation" level, respectively. The *domain* is a finite set that enumerates the possible values that can be taken by resources according to the rule's topic. For instance, a subset of domain for a rule dealing with the right topic is {"no-retention", "limited-use"}. The *scope* of a rule defines the granularity of the resource that is subject to privacy constraints. We consider two cases: operation and data rules. In the former case, several parts of a service log entry may be viewed as private. Services assign one of the values "total" or "partial" to the scope of their operation resources. If an operation resource is assigned a "total" scope for a given rule, then the whole entry of that operation in the service log is private. Otherwise (i.e., the assigned scope is "partial"), only the ID of the service that invoked the operation is private. In the case of data rules, we consider data resources as atomic. Hence, the only scope value allowed in this situation is {"total"}. "Partial" scope may also be considered for complex data resources (e.g., array structure). In this case, only part of an input/output parameter is private. However, this issue is out of the scope of this paper. Two rules at most are created for each topic: one for data and another for operations.

Definition 2. A privacy rule R_i is defined by a tuple (T_i, L_i, D_i, S_i) where:

- T_i is the topic of R_i,
- $L_i \in$ {"data", "operation"} is the level of the rule,
- D_i is the domain set of R_i; it enumerates the possible values that can be taken by T_i with respect to *rs*,
- S_i is the scope of R_i where S_i= {"total","partial"} if L_i="operation" and S_i={"total"} if L_i="data". ◊

For instance, we give two examples of rules R_1 and R_2, where R_1 = (T_1, L_1, D_1, S_1) with T_1="recipient", L_1="data", D_1= {`public`, `government` , `federal tax`, `research` } and S_1 ={"total"} R_2 = (T_2, L_2, D_2, S_2) with T_2="recipient", L_2="operation", D_1= {`public`} and S_2 ={"total", "partial"}. Our objective is to propose formal privacy model with a fine granularity that allows to add, modify (e.g., add new topic) and delete rules at anytime but also to check formally the compatibility between rules among service. It is therefore important to examine how privacy rules can be instantiated which is the focus of the subsequent section.

3.3 Privacy Assertion

The services will use privacy rules to define the privacy features of their resources. The application of a rule R_i=(T_i, L_i, D_i, S_i) on a resource rs is a *privacy assertion* $A(R_i, rs)$ where rs has L_i as a level. $A(R_i, rs)$ states the granularity of rs that is subject to privacy. The granularity g belongs to the scope S_i of the rule. For instance, g is equal to partial if only the ID of the operation invoker is private. $A(R_i, rs)$ also indicates D_i's values that are attributed to rs. For example, let us consider the rule R_1. A privacy assertion on rs according to R_1 may state that rs will be shared with government agencies and research institutions. We use the *propositional formula* (pf)"`government`" $\wedge$ "`research`" to specify such statement.

Definition 3. A privacy assertion $A(R_i, rs)$ on a resource rs is defined by the couple (pf, g); $pf = v_{ip} \wedge ... \wedge v_{iq}$ according to R_i=(T_i, L_i, D_i, S_i), where v_{ip},..., $v_{iq} \in D_i$; $g \in S_i$ is the granularity of rs. ◊

3.4 Privacy Policy

A service S will define a *privacy policy*, $\texttt{PP}^{\texttt{S}}$, that specifies the set of practices applicable to the collected resources. Each service has its own perception of what should be considered as private. Defining the privacy policy $\texttt{PP}^{\texttt{S}}$ of S is performed in two steps. First, the service S identifies the set (noted P_p) of all privacy resources. Second, S specifies assertions for each resource rs in P_p. Deciding about the content of P_p and the rules (from RS) to apply to each resource in P_p varies from a service to another. $\texttt{PP}^{\texttt{S}}$ specifies the way S (i) treats the collected resources (i.e., received through the mediator), (ii) expects any third-party services to treat resources provided as output when S operation will be invoked. We consider three cases: (a) rs is an input data, (b) rs is an output data, and (c) rs is an operation. If rs is an input data or operation (cases (a) and (c)), then $A(R_i, rs)$ states what will a service S do with rs according to R_i. If rs is an output data (case (b)), then S defines two assertions for rs according to R_i; the first, noted $A(R_i, rs^E)$, gives S's expectation; the second, $A(R_i, rs^P)$, denotes S's practice:

- Expectation: $A(R_i, rs^E)$ states what service S expects a third-party service to do with rs (provided as the output of S operation) according to R_i.
- Practice: $A(R_i, rs^P)$ states what service S will do with rs according to R_i.

For instance, let us consider a scientist that would like to conduct some experiments. Through mediator, the operation op_b of the service S_b will be invoked. op_b takes as input a `patient-disease` and returns as output the `SSN` (social security number) of the patient. The service S_b (which owns operation op_b) expects that third-party services will use the given output of op_b according to its expectations since `SSN` is a data with higher privacy sensitivity. We give below a definition of privacy policy and rs_k refers rs_k^E or rs_k^P if rs_k is an output data.

Definition 4. The *privacy policy* of a service S is defined as $\texttt{PP}^{\texttt{S}} = \{A_j(R_i, rs_k), j \leqslant |\texttt{PP}^{\texttt{S}}|, i \leqslant |RS|, k \leqslant |P_p|, rs_k \in RS\}$

3.5 Privacy Requirements

A service S will define a *Privacy Requirements* $\mathtt{PR}^{\mathtt{S/T}}$ stating S's assertions describing how S expects and requires a third-party service T should use its resources. Before creating $\mathtt{PR}^{\mathtt{S/T}}$, S first identifies the set (noted P_c) of all its privacy-sensitive resources. $\mathtt{PR}^{\mathtt{S/T}}$ assertions describe the following requirements:

- The way S expects T to treat the privacy of input data, output data (e.g., experiment results returned by a service), and information about operation invocation; and
- The way S treats the privacy of any output data returned by T, through the mediator.

The aforementioned requirements are expressed via privacy assertions. Similarly to privacy policies, requirements on outputs express service's expectations (noted $A(R_i, rs^E)$) and practices (noted $A(R_i, rs^P)$). For instance, the output of operation invoked (owned by a third-party service) by S concerns primary S and S may be sensitive about how third-party service owned the invoked operation, will treat the output of the invoked operation regarding retention time. S may unequally value the assertions specified in $\mathtt{PR}^{\mathtt{S/T}}$. For instance, S owns SSN and zip_code data, S's requirements about SSN may be stronger than its requirements for zip_code. Besides, S may consider an assertion more essential than another, even if both assertions are about the same resource. For example, S may view the rule constraining the recipients of SSN as more valuable than the rule stating the duration for which the service can retain SSN. For that purpose, S assigns a weight w_j to each assertion $A(R_i, rs)$ in $\mathtt{PR}^{\mathtt{S/T}}$. w_j is an estimate of the significance of $A(R_i, rs)$. The higher is the weight, the more important is the corresponding assertion. Each weight is decimal number between 0 and 1. The total of weights assigned to all assertions equals 1:

- $\forall\ j \leqslant |\mathtt{PR}^{\mathtt{S/T}}| : 0 < w_j \leqslant 1$,
- $\sum_{j=1}^{k} w_j = 1$, where $k = |\mathtt{PR}^{\mathtt{S/T}}|$

In the real cases, the service S may be willing to update some of their privacy requirements. For instance, it may agree to relax constraints about the disclosure of their zip_code if the mediator requests that in exchange to offer it incentives such as discounts. However, S will probably be more reluctant to loosen conditions about the disclosure of their names. To capture this aspect, S stipulates whether an assertion $A(R_i, rs)$ is *mandatory* or *optional* via a boolean attribute M_j attached to assertion$A$.

Definition 5. The *privacy requirements* of a service S on third service T is defined as $\mathtt{PR}^{\mathtt{S/T}}$= { $(A_j(R_i, rs_k), w_j, M_j)$, $j \leqslant |\mathtt{PR}^{\mathtt{S/T}}|$, $i \leqslant |RS|$, $k \leqslant |P_c|$, $rs_k \in P_c$, $R_i \in RS$, w_j is the weight of A_j, M_j=True iff A_j is mandatory }. ◊

Other specific conditions, related to the context application, may be specified with SPARQL conditions (as showed in Figure 1). Furthermore, services may use

privacy protection mechanism (like k-anonymity) to sanitize its data(Figure 1). Due to the space limitation, details of these two characteristics will be discussed in another future work.

4 Privacy Compatibility

In this section we introduce the notion of compatibility between privacy policies and requirements according the notion of privacy subsumption.

4.1 Privacy Subsumption

Let us consider a rule R_i=(T_i, L_i, D_i, S_i). Defining an assertion $A(R_i, rs)$=(pf, g) for rs involving assigning value(s) from D_i to the propositional formula pf of A. The values in D_i are related to each other. For instance, let us consider the domain {`public`, `government`, `federal tax`, `research`} for a rule dealing with topic T_i="recipient". The value `public` is more general than the other values in D_i. Indeed, if the recipient of rs is declared public (i.e., shared with any entity), then the recipient is also government and research. Likewise, the value `government` is more general than `research` since the research **is-a** government agency. To capture the semantic relationship among domain values, we introduce the notion of *privacy subsumption* (noted $\sqsubseteq$). For instance, the following subsumptions can be stated: `government` $\sqsubseteq$ `public`; `research` $\sqsubseteq$ `government`. Note that privacy subsumption is transitive since it models the "is-a" relationship. We use $*$ to refer to the transitive closure of $\sqsubseteq$.

Definition 6. Let $D_i = \{v_{i1}, ..., v_{im}\}$ be the domain of a privacy rule R_i. We say that v_{ip} is subsumed by v_{iq} or v_{iq} subsumes v_{ip}, ($1 \leqslant p \leqslant m$ and $1 \leqslant q \leqslant m$) noted $v_{ip} \sqsubseteq v_{iq}$, iff v_{iq} is more general than v_{ip}. ◊

We generalize the notion of privacy subsumption to assertions. Let us consider an assertion $A(R_i, rs)$=(pf, g) representing an expectation of `S` (resp., `T`) and another assertion $A'(R_i^{'}, rs')$=(pf', g') modeling a practice of `T` (resp., `S`). In order for A and A' to be compatible, they must be specified on the same rule (R_i=$R_i^{'}$), the same resource (rs=rs'), and at the same granularity (g=g'). Besides, the expectation of `S` (resp., `T`) as stated by pf should be more general (i.e., subsumes) than the practice of `S` (resp., `T`) as given by pf'. In other words, if pf is true, then pf' should be true as well. For instance, if pf="`government` $\wedge$ `research`" and pf'='`government`", then $pf \Rightarrow pf'$ (where $\Rightarrow$ is the symbol for implication in propositional calculus). Hence, A is more general than A' or A subsumes A' (noted $A' \sqsubseteq A$).

Although some literals used in pf are syntactically different from the ones used in pf', they may be semantically related via subsumption relationships. For instance, let us assume that pf="`public` $\wedge$ `research`" and pf'="`federal tax`". Since `federal tax` $\sqsubseteq$ `public`, we can state that `public` $\Rightarrow$ `federal tax`. In this case, we can prove that $pf \Rightarrow pf'$ and hence, $A' \sqsubseteq A$. To deal with the issue of having different literals in propositional formulas, we use the following

property: if $v_{ip} * v_{iq}$ (i.e., v_{iq} directly or indirectly subsumes v_{ip}) then $v_{iq} \Rightarrow v_{ip}$.

Definition 7. Let us consider assertions $A(R_i, rs)=(pf, g)$ and $A'(R_i^{'}, rs')=(pf', g')$. A' is subsumed by A or A subsumes A', noted $A' \sqsubseteq A$, if $R_i=R_i^{'}$, $rs=rs'$, $g=g'$, and $pf \Rightarrow pf'$. ◊

4.2 Privacy Compatibility Matching Algorithm

The aim of Privacy Compatibility Matching algorithm *PCM* is to check that assertions in $\texttt{PR}^{\texttt{S/T}}$ and $\texttt{PP}^{\texttt{T}}$ are related via subsumption relationships (cf. Definition 7). As mentioned in 3.2 and 3.3, both $\texttt{PR}^{\texttt{S/T}}$ and $\texttt{PP}^{\texttt{T}}$ contain expectations and practices. *PCM* matches expectations in $\texttt{PR}^{\texttt{S/T}}$ to practices in $\texttt{PP}^{\texttt{T}}$ and expectations in $\texttt{PP}^{\texttt{T}}$ to practices in $\texttt{PR}^{\texttt{S/T}}$. *PCM* deals with the following three cases:

Case (a) *PCM* matches a $\texttt{PR}^{\texttt{S/T}}$ assertion $A(R_i, rs)$ where rs is an input or operation usage, to an assertion $A'(R_i^{'}, rs')$ in $\texttt{PP}^{\texttt{T}}$. In this case, $A(R_i, rs)$ is a S's expectation and $A'(R_i^{'}, rs')$ is a $\texttt{PP}^{\texttt{T}}$ practice. If $A' \sqsubseteq A$ then A' and A are matched.

Case (b) *PCM* matches a $\texttt{PR}^{\texttt{S/T}}$ assertion $A(R_i, rs^E)$ where rs^E is an output, to an assertion $A'(R_i^{'}, rs'^P)$ in $\texttt{PP}^{\texttt{T}}$. In this case, $A(R_i, rs^E)$ is a S's expectation and $A'(R_i^{'}, rs'^P)$ is a $\texttt{PP}^{\texttt{T}}$ practice. If $A' \sqsubseteq A$ then A' and A are matched.

Case (c) *PCM* matches a $\texttt{PR}^{\texttt{S/T}}$ assertion $A(R_i, rs^P)$ where rs^P is an output, to an assertion $A'(R_i^{'}, rs'^E)$ in $\texttt{PP}^{\texttt{T}}$. In this case, $A(R_i, rs^P)$ is a S's expectation and $A'(R_i^{'}, rs'^E)$ is a $\texttt{PP}^{\texttt{T}}$ practice. If $A' \sqsubseteq A$ then A' and A are matched.

Two options are possible while matching $\texttt{PR}^{\texttt{S/T}}$ and $\texttt{PP}^{\texttt{T}}$. The first option is to require full matching. This is not flexible since some DaaS consumers may be willing to use a DaaS producer even if certain of their privacy constraints are not satisfied. For that purpose, we present a *cost model*-based solution to enable *partial matching*. The cost model combines the notions of *privacy matching degree* and *threshold*. Due to the large number and heterogeneity of DaaS services, it is not always possible to find policy $\texttt{PP}^{\texttt{T}}$ that fully matches a S's requirement $\texttt{PR}^{\texttt{S/T}}$. The *privacy matching degree* gives an estimate about the ratio of $\texttt{PR}^{\texttt{S/T}}$ assertions that are matched to $\texttt{PP}^{\texttt{T}}$ assertions. We refer to $\mathbf{m} \subset \texttt{PR}^{\texttt{S/T}}$ as the set of all such $\texttt{PR}^{\texttt{S/T}}$ assertions. The degree is obtained by adding the weights of all assertions in $\mathbf{m}$: $\text{Degree}(\texttt{PR}^{\texttt{S/T}}, \texttt{PP}^{\texttt{T}}) = \sum w_j$ for all assertions $(A_j(R_i, rs_k), w_j, M_j) \in \mathbf{m}$. The privacy matching *threshold* τ gives the minimum value allowed for a matching degree. The value of τ is given by the client and gives an estimate of how much privacy the consumer is willing to sacrifice. As mentioned in 3.5, we give consumer the possibility to control their "core" privacy requirements by associating a mandatory attribute M_j to each assertion $(A_j(R_i, rs_k), w_j, M_j)$ in $\texttt{PR}^{\texttt{S/T}}$.

5 Privacy-Aware DaaS Composition

We aims at extending the composition approach described in [2] to deal with privacy preserving according three step. First, a functional selection of DaaS is performed. taking as input a user query. Second, the privacy requirements and policy attached to service are fetched, thanks to the annotation approach developed in [14]. Third, the compatibility of privacy requirements and policies of the services with respect to the composition is checked. In this section we give details about these steps.

5.1 Fetching DaaS Annotations

The model developed above provides a formal background to specify privacy requirements $\mathtt{PR}^{\mathtt{S/T}}$ and policy $\mathtt{PP}^{\mathtt{S}}$ of service `S`. To make these privacy capabilities concretely available on service, we link them to services via an annotation of the service. Our previous work in [14] provided a complete description about the privacy annotation extensibility. We remind how we annotate the major description formats for DaaS (WSDL and REST annotations) according to the aforementioned privacy model. Indeed, the specifications of WSDL allows for the addition of new XML elements and attributes in certain locations inside a WSDL file. We exploit these extensibility elements to associate the services operations, interface inputs and outputs with their corresponding capability files. Specifically, for each interface, operation, input and output elements, we define a new child element called "`privacy-capability`" to hook assertions of $\mathtt{PR}^{\mathtt{S/T}}$ and assertions of $\mathtt{PR}^{\mathtt{S}}$ with to `S` descriptions. For retro-compatibility sake, we also provide the following rules to adapt our WSDL 2.0 annotation to WSDL 1.1. The "attrExtensions" element defined in SAWSDL are utilized to annotate elements that do not support attribute extensibility, such as `operation` and `porttype`. The `porttype` element must be annotated as the ancestor of the `interface` WSDL 2.0 element, and message `part` elements must be annotated in replacement of `input` and `output` WSDL 2.0 elements. During the composition, the privacy requirement description file of component service is compared to this describing the privacy policy of service within composition as explaining in the above subsection.

5.2 Checking Privacy within Composition

We aims at extending the previous composition approach to deal with privacy preserving. Let us consider services in Table 1 and the following epidemiologist's query Q "What are Ages, Genders, address, DNA, salaries of patients infected with *H1N1*; and what are the global weather conditions of the area where these patients reside?". The mediator is considered as a trusted entity. It manage the composition and handles all the interactions among services. It answer Q by composing the relevant services as follows: Firstly, the invocation of $S_{1.1}$ with *H1N1*, then for each obtained patient, $S_{4.1}$ is invoked to obtain their `DNA`, $S_{2.2}$ and $S_{3.1}$ to obtain `date_of_birth`, `zip_code` and `salary` of obtained patients. Finally, $S_{5.1}$ with patients'`zip_code` to get information about

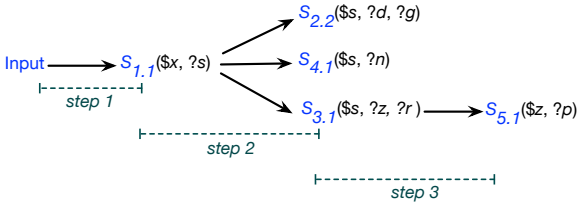


Fig. 2. Dependency Graph of query Q

the `weather_conditions`. Selected services need to be executed in a *particular order* in the composition plan depending on their inputs and outputs. To construct the composition plan the algorithm establishes a dependency graph *DG* (Figure 2) in which the nodes correspond to services and the edges correspond to dependency constraints between component services. If a service S_j need an input x that can be provided from an output y of S_i then S_j must be preceded by S_i in the composition plan; we say that there is a *dependency* between S_i and S_j (or S_j depends on S_i). Consequently, the mediator recognizes that services $S_{2.2}$, $S_{3.1}$, $S_{4.1}$ depend on $S_{1.1}$ since they have a same input y (i.e. `SSN`) which is provided as an output of $S_{1.1}$ and $S_{5.1}$ depends on $S_{3.1}$.

In order to take privacy into account, if S_j depends on S_i, then S_j is showed as a consumer to some data provided by S_i and this latter is showed then as a producer from the mediator point of view. In other words, the mediator considers the privacy requirements $\texttt{PR}^{S_i/\texttt{T}}$ for service S_i (since $\texttt{PR}^{S_i/\texttt{T}}$ specifies S_i' conditions on the usage of its concerning data) and privacy policy $\texttt{PP}^{S_j}$ for service S_j (since $\texttt{PP}^{S_j}$ specifies S_j' usage on the collected data) and checks the compatibility of $\texttt{PR}^{S_i/\texttt{T}}$ and $\texttt{PP}^{S_j}$ by using the privacy compatibility matching algorithm *PCM* (Section 4.2) within services order in *DG*.

For instance, let us consider *DG* in Figure 2. The mediator identifies firstly, from *DG*, services type (i.e., consumer services, and producer services) and resources related to each dependency. The parameter s is an input parameter for the services $S_{2.2}$, $S_{3.1}$ and $S_{4.1}$ while it is an output parameter for $S_{1.1}$ and therefore $S_{2.2}$, $S_{3.1}$ and $S_{4.1}$ depend on $S_{1.1}$. Note that input parameters begins with "$" and output parameters by "?". Similarly, the parameter z is an input parameter for $S_{5.1}$ and an output parameter for $S_{3.1}$, therefore $S_{5.1}$ depends on $S_{3.1}$. Consequently, mediator considers $S_{2.2}$, and $S_{4.1}$ as consumers services, while $S_{1.1}$ is considered once as a consumer (since it input is provided by the *input*) once as a producer (since it provides output for others services). The same reasoning is observed for $S_{3.1}$. In *step 1* the producer is the *input* (i.e., the user query), consumer is $S_{1.1}$ and the private resource rs ="`Patient Disease`". The mediator checks the compatibility of $\texttt{PR}^{input/\texttt{T}}$ and $\texttt{PP}^{S_{1.1}}$. In *step 2* the producer is $S_{1.1}$ and consumers are $S_{2.2}$, $S_{3.1}$, $S_{4.1}$ and the private resource rs ="`SSN`". The mediator checks the compatibility of $\texttt{PR}^{S_{1.1}/\texttt{T}}$ and $\texttt{PP}^{S_{2.2}}$, $\texttt{PR}^{S_{1.1}/\texttt{T}}$ and $\texttt{PP}^{S_{3.1}}$, $\texttt{PR}^{S_{1.1}/\texttt{T}}$ and $\texttt{PP}^{S_{4.1}}$, In *step 3* $S_{3.1}$ is now the producer for $S_{5.1}$ and rs ="`zip_code`" and the compatibility of $\texttt{PR}^{S_{3.1}/\texttt{T}}$ and $\texttt{PP}^{S_{5.1}}$ is checked.

Table 1. A subset of PAIRSE's DaaSs

DaaS services	Semantics services Description
$S_{1.1}(\$x, ?s)$ $S_{1.2}(\$x, ?s)$	Returns patients SNN s, infected with a disease x
$S_{2.1}(\$s, ?d, ?g)$ $S_{2.2}(\$s, ?d, ?g)$	Returns date_of_birth, d, and gender, g, of a patient identified by s
$S_{3.1}(\$s, ?z, ?r)$	Returns zip_code, z, and salary, r, of a patient identified by s
$S_{4.1}(\$s, ?n)$ $S_{4.2}(\$s, ?n)$	Returns DNA, n, of a patient identified by s
$S_{5.1}(\$z, ?w)$	Returns Weather_condition, w, of a address z

6 Evaluations

To demonstrate the feasibility of our approach to privacy-preserving DaaS composition, we applied it to a real scenario drawn from the healthcare domain. In the context of the PAIRSE project [2] we were provided with access to /411/ medical Web services defined on top of /23/ different medical databases (oracle databases) storing medical information (e.g., diseases, medical tests, allergies, etc) about more than /30,000/ patients. Among these services Table1 shows the services that pertain to the query Q in our running example. All services were deployed on top of a GlassFish web server. The resources are related to a particular type of medical data (e.g., ongoing treatments, Allergies). For each service, we have randomly generated privacy requirements and privacy policy with regard to /10/ values D_i set for R_i topic = "medical recipients" (e.g., researcher, physician, nurse, etc) and different values for R_i topic = "purpose" (e.g., scientific research, academic laboratory, government, etc.). These privacy requirements and policies are used to annotate the service description files in accordance with the mechanisms presented in section 5. Our algorithms are implemented in Java and run on a Intel Core Duo 2.53 GHz and 4GB RAM running Windows 7.

Table 2. Possible compositions that answer Q without and with privacy preservation

Compositions without privacy preservation	Compositions with privacy preservation
$C_1 = \{S_{1.1}, S_{2.1}, S_{3.1}, S_{4.1}, S_{5.1}\}$	$C_3 = \{S_{1.1}, S_{2.2}, S_{3.1}, S_{4.1}, S_{5.1}\}$
$C_2 = \{S_{1.1}, S_{2.1}, S_{3.1}, S_{4.2}, S_{5.1}\}$	$C_4 = \{S_{1.1}, S_{2.2}, S_{3.1}, S_{4.2}, S_{5.1}\}$
$C_3 = \{S_{1.1}, S_{2.2}, S_{3.1}, S_{4.1}, S_{5.1}\}$	
$C_4 = \{S_{1.1}, S_{2.2}, S_{3.1}, S_{4.2}, S_{5.1}\}$	
$C_5 = \{S_{1.2}, S_{2.1}, S_{3.1}, S_{4.1}, S_{5.1}\}$	
$C_6 = \{S_{1.2}, S_{2.1}, S_{3.1}, S_{4.2}, S_{5.1}\}$	
$C_7 = \{S_{1.2}, S_{2.2}, S_{3.1}, S_{4.1}, S_{5.1}\}$	
$C_8 = \{S_{1.2}, S_{2.2}, S_{3.1}, S_{4.2}, S_{5.1}\}$	

[2] https://picoforge.int-evry.fr/cgi-bin/twiki/view/Pairse/Web/

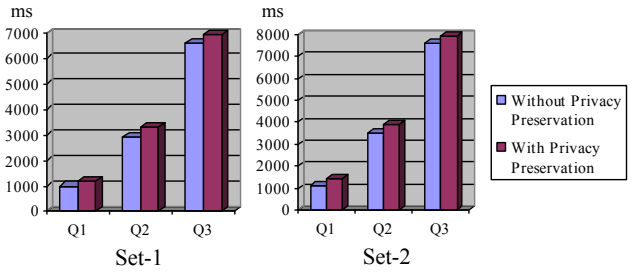


Fig. 3. The Experimental Results

Table 2 shows in the first column the different DaaSs compositions the composition would give without applying our privacy compatibility matching algorithm *PCM*. Much of these composition may violate the privacy requirements of involved services. The second column shows the possible compositions when *PCM* within composition approach (of section 5.1) is applied. These compositions do preserve the privacy requirements of involved services. We conducted a set of experiments to measure the cost incurred in privacy preservation while composing DaaS. We considered two sets of queries. The first one included queries about a given patient, each with a different size: Q_1 requests the "Personal information" of a given patient p_i, Q_2 requests the "Personal information", "Allergies" and "Ongoing Treatments" of p_i, and Q_3 requests the "Personal information", "Allergies", "Ongoing Treatments", "Cardiac Conditions" and "Biological Tests" of p_i. The second set uses the same queries Q_1, Q_2 and Q_3 but for all of patients living in Lyon. All queries were posed by the same actor (researcher) and for the same purpose (medical research). Figure 3 depicts the results obtained for the queries in sets 1 and 2,(the time shown includes both the DaaS composition construction time and the DaaS composition execution time). Set-2 (as opposed to Set-1) amplifies the cost incurred by Set-1 at the composition "*execution phase*" by a factor equals to the number of returned patients. The results for Set-1 show that privacy handling adds only a slight increase in the query rewriting time (note that the composition execution time is neglected for one patient). This is due to the fact that the number of services used to retrieve privacy requirements is limited compared to the number of services used to retrieve data (10 versus 411 in our experiments). The results for Set-2 show that the extra time needed to handle privacy in the the composition process is still relatively low if compared to the time required for answering queries without addressing privacy concerns.

7 Conclusion

In this paper, we proposed a dynamic and formal privacy model for DaaS services. The model deals with privacy at two different levels: the data (inputs and outputs) and operation levels. Services specify their privacy concerns/practices via privacy requirements and policies, respectively. Both privacy requirements

and policies refer to rules that may be added, deleted, and modified at any time. The granularity of our privacy model allows defining the widest range of policies and requirement with rich expression capabilities and flexibly manner. We introduced a cost model-based protocol for checking the compatibility of privacy requirements and policies. We have presented a preserving-privacy DaaS composition approach to resolve privacy concerns at the composition time. As future work, we plan to extend our privacy-preserving DaaS composition approach to tackle the incompatibilities between requirements and policies using a dynamic reconciliation mechanism. The reconciliation of requirements and policies will be carried out based on some negotiation protocols. We intend also to study and improve the scalability of our proposed privacy-aware composition approach.

Acknowledgment. This work has partially funded by the Region of Rhône-Alpes.

References

1. Agrawal, S., Haritsa, J.R.: A framework for high-accuracy privacy-preserving mining. In: Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, pp. 193–204. IEEE Computer Society, Washington, DC, USA (2005)
2. Barhamgi, M., Benslimane, D., Medjahed, B.: A Query Rewriting Approach for Web Service Composition. IEEE Transactions on Services Computing, TSC (January 2010)
3. Bertino, E., Yang, Y.: Privacy and ownership preserving of outsourced medical data. In: ICDE, pp. 521–532 (2005)
4. Carey, M.: Declarative data services: This is your data on soa. In: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications, p. 4. IEEE Computer Society, Washington, DC, USA (2007)
5. Clifton, C., Kantarcioğlu, M., Doan, A., Schadow, G., Vaidya, J., Elmagarmid, A., Suciu, D.: Privacy-preserving data integration and sharing. In: DMKD 2004: Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 19–26. ACM, New York (2004)
6. Feder, T., Ganapathy, V., Garcia-Molina, H., Motwani, R., Thomas, D.: Distributing data for secure database services. Technical Report 2007-23, Stanford InfoLab (June 2007)
7. Gil, Y., Cheung, W., Ratnakar, V., kin Chan, K.: Privacy enforcement in data analysis workflows. In: Finin, T., Kagal, L., Olmedilla, D. (eds.) Proceedings of the Workshop on Privacy Enforcement and Accountability with Semantics (PEAS 2007) at ISWC/ASWC 2007, Busan, South Korea. CEUR Workshop Proceedings, vol. 320. CEUR-WS.org (November 2007)
8. Gil, Y., Fritz, C.: Reasoning about the appropriate use of private data through computational workflows. In: Intelligent Information Privacy Management, Papers from the AAAI Spring Symposium, pp. 69–74 (March 2010)
9. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, SIGMOD 2002, pp. 216–227. ACM, New York (2002)

10. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, pp. 720–731. VLDB Endowment (2004)
11. Kawamoto, J., Yoshikawa, M.: Security of social information from query analysis in daas. In: Proceedings of the 2009 EDBT/ICDT Workshops, EDBT/ICDT 2009, pp. 148–152. ACM, New York (2009)
12. Lee, Y., Werner, J., Sztipanovits, J.: Integration and verification of privacy policies using DSML's structural semantics in a SOA-based workflow environment. Journal of Korean Society for Internet Information 10(149), 09/2009 (2009)
13. Mohammed, N., Fung, B.C.M., Wang, K., Hung, P.C.K.: Privacy-preserving data mashup. In: EDBT 2009: Proceedings of the 12th International Conference on Extending Database Technology, pp. 228–239. ACM, New York (2009)
14. Mrissa, M., Tbahriti, S.-E., Truong, H.-L.: Privacy model and annotation for DaaS. In: Antonio Brogi, G.A.P., Pautasso, C. (eds.) European Conference on Web Services (ECOWS), pp. 3–10 (December 2010)
15. Ngu, A.H.H., Carlson, M.P., Sheng, Q.Z., Paik, H.-y.: Semantic-based mashup of composite applications. IEEE Trans. Serv. Comput. 3, 2–15 (2010)
16. Pang, H., Shen, J., Krishnan, R.: Privacy-preserving similarity-based text retrieval. ACM Trans. Internet Technol. 10, 4:1–4:39 (2010)
17. Truong, H.L., Dustdar, S.: On analyzing and specifying concerns for data as a service. In: Kirchberg, M., Hung, P.C.K., Carminati, B., Chi, C.-H., Kanagasabai, R., Valle, E.D., Lan, K.-C., Chen, L.-J. (eds.) APSCC, pp. 87–94. IEEE, Los Alamitos (2009)
18. Tumer, A., Dogac, A., Toroslu, I.H.: A semantic-based user privacy protection framework for web services. In: Mobasher, B., Anand, S.S. (eds.) ITWP 2003. LNCS (LNAI), vol. 3169, pp. 289–305. Springer, Heidelberg (2005)
19. W3C. The Platform for Privacy Preference Specification (2004)
20. Weise, T., Bleul, S., Comes, D., Geihs, K.: Different approaches to semantic web service composition. In: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services, pp. 90–96. IEEE Computer Society, Washington, DC, USA (2008)
21. Xu, Y., Wang, K., Zhang, B., Chen, Z.: Privacy-enhancing personalized web search. In: Proceedings of the 16th international conference on World Wide Web, WWW 2007, pp. 591–600. ACM, New York (2007)

An Empirical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities

Su Zhang, Doina Caragea, and Xinming Ou

Kansas State University
{zhangs84,dcaragea,xou}@ksu.edu

Abstract. Software vulnerabilities represent a major cause of cyber-security problems. The National Vulnerability Database (NVD) is a public data source that maintains standardized information about reported software vulnerabilities. Since its inception in 1997, NVD has published information about more than 43,000 software vulnerabilities affecting more than 17,000 software applications. This information is potentially valuable in understanding trends and patterns in software vulnerabilities, so that one can better manage the security of computer systems that are pestered by the ubiquitous software security flaws. In particular, one would like to be able to predict the likelihood that a piece of software contains a yet-to-be-discovered vulnerability, which must be taken into account in security management due to the increasing trend in zero-day attacks. We conducted an empirical study on applying data-mining techniques on NVD data with the objective of predicting the time to next vulnerability for a given software application. We experimented with various features constructed using the information available in NVD, and applied various machine learning algorithms to examine the predictive power of the data. Our results show that the data in NVD generally have poor prediction capability, with the exception of a few vendors and software applications. By doing a large number of experiments and observing the data, we suggest several reasons for why the NVD data have not produced a reasonable prediction model for time to next vulnerability with our current approach.

Keywords: data mining, cyber-security, vulnerability prediction.

1 Introduction

Each year a large number of new software vulnerabilities are discovered in various applications (see Figure 1). Evaluation of network security has focused on known vulnerabilities and their effects on the hosts and networks. However, the potential for unknown vulnerabilities (*a.k.a.* zero-day vulnerabilities) cannot be ignored because more and more cyber attacks utilize these unknown security holes. A zero-day vulnerability could last a long period of time (*e.g.* in 2010 Microsoft confirmed a vulnerability in Internet Explorer, which affected some

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 217–231, 2011.

versions that were released in 2001). Therefore, in order to have more accurate results on network security evaluation, one must consider the effect from zero-day vulnerabilities. The National Vulnerability Database (NVD) is a well-known data source for vulnerability information, which could be useful to estimate the likelihood that a specific application contains zero-day vulnerabilities based on historical information. We have adopted a data-mining approach in an attempt to build a prediction model for the attribute "time to next vulnerability" (TTNV), *i.e.* the time that it will take before the next vulnerability about a particular application will be found. The predicted TTNV metrics could be translated into the likelihood that a zero-day vulnerability exists in the software.

Past research has addressed the problem of predicting software vulnerabilities from different angles. Kyle *et al.* [10] pointed out the importance of estimating the risk-level of zero-day vulnerabilities. Mcqueen *et al.* [15] did experiments on estimating the number of zero-day vulnerabilities on each given day. Alhazmi and Malaiya [3] introdced the definition of TTNV. Ozment [19] did a number of studies on analyzing NVD, and pointed out several limitations of this database.

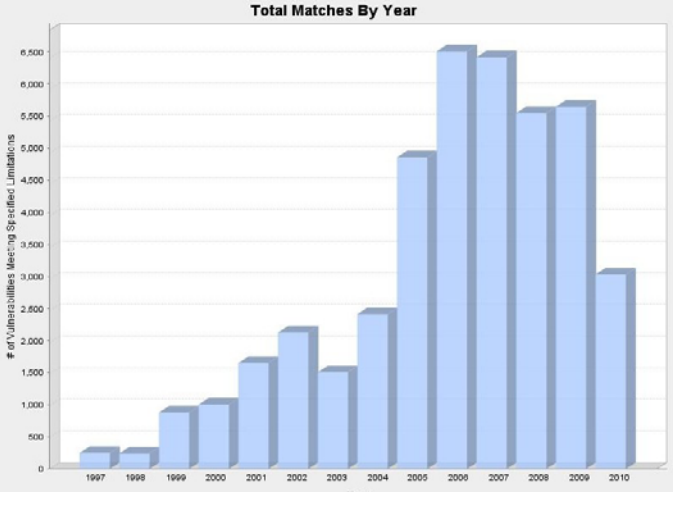


Fig. 1. The trend of vulnerability numbers

In this paper, we present our empirical experience of applying data-mining techniques on NVD data in order to build a prediction model for TTNV. We conduct a rigorous data analysis and experiment with a number of feature construction schemes and learning algorithms. Our results show that the data in NVD generally have poor prediction capability, with the exception of a few vendors and software applications. In the rest of the paper we will explain the features we have constructed and the various approaches we have taken in our attempts to build the prediction model. While it is generally a difficult task to show that data has no utility, our experience does indicate a number of reasons

why it is unlikely to construct a reliable prediction model for TTNV given the information available in NVD.

2 Data Source – National Vulnerability Database

Each data entry in NVD consists of a large number of fields. We represent them as <D, CPE, CVSS>. D is a set of data including published time, summary of the vulnerability and external links about each vulnerability. CPE [6] and CVSS [21] will be described below.

2.1 CPE (Common Platform Enumeration)

CPE is *an open framework for communicating the characteristics and impacts of IT vulnerabilities.* It provides us with information on a piece of software, including version, edition, language, *etc.* An example is shown below:

```
cpe:/a:acme:product:1.0:update2:pro:en-us
    Professional edition of the "Acme Product 1.0 Update 2 English".
```

2.2 CVSS (Common Vulnerability Scoring System)

CVSS is a vulnerability scoring system designed to provide an open and standardized method for rating IT vulnerabilities. CVSS helps organizations prioritize and coordinate a joint response to security vulnerabilities by communicating the base, temporal and environmental properties of a vulnerability. Currently NVD only provides the Base group in its metric vector. Some components of the vector are explained below.

- *Access Complexity* indicates the difficulty level of the attack required to exploit the vulnerability once an attacker has gained access to it. It includes three levels: High, Medium, and Low.
- *Authentication* indicates whether an attacker must authenticate in order to exploit a vulnerability. It includes two levels: Authentication Required (R), and Authentication Not Required (NR).
- *Confidentiality, Integrity and Availability* are three loss types of attacks. Confidentiality loss means information will be leaked to people who are not supposed to know it. Integrity loss means the data can be modified illegally. Availability loss means the compromised system cannot perform its intended task or will crash. Each of the three loss types have the three levels: None (N), Partial (P), and Complete (C).

The *CVSS Score* is calculated based on the metric vector, with the objective of indicating the severity of a vulnerability.

3 Our Approach

We choose TTNV (time to next vulnerability) as the predicted feature. The predictive attributes are time, versiondiff (the distance between two different versions by certain measurement), software name and CVSS. All are derived or extracted directly from NVD.

3.1 Data Preparation and Preprocessing

Division of training/test data: As the prediction model is intended to be used to forecast future vulnerabilities, we divide the NVD data into training and test data sets based on the time the vulnerabilities were published. The ratio of the amount of training to test data is 2. We chose to use the data starting from 2005, as the data prior to this year looks unstable (see Figure 1).

Removing obvious errors: Some NVD entries are obviously erroneous (*e.g.* in one entry for Linux the kernel version was given as 390). To prevent these entries from polluting the learning process, we removed them from the database.

3.2 Feature Construction and Transformation

Identifying and constructing predictive features is of vital importance to data-mining. For the NVD data, intuitively Time and Version are two useful features. As we want to predict time to next vulnerability, the published time for each past vulnerability will be a useful information source. Likewise, the version information in each reported vulnerability could indicate the trend of vulnerability discovery, as new versions are released. Although both Time and Version are useful information sources, they need to be transformed to provide the expected prediction behavior. For example, both features in their raw form increase monotonically. Directly using the raw features will provide little prediction capability for future vulnerabilities. Thus, we introduce several feature construction schemes for the two fields and studied them experimentally.

Time: We investigated two schemes for constructing time features. One is epoch time, the other is using month and day separately without year. Like explained before, the epoch time is unlikely to provide useful prediction capability, as it increases monotonically. Intuitively, the second scheme shall be better, as the month and day on which a vulnerability is published may show some repeating pattern, even in future years.

Version: We calculate the difference between the versions of two adjacent instances and use the versiondiff as a predictive feature. An instance here refers to an entry where a specific version of an application contains a specific vulnerability. The rationale for using versiondiff as a predictive feature is that we want to use the trend of the versions with time to estimate future situations. Two versiondiff schemas are introduced in our approach. The first one is calculating the versiondiff based on version counters (rank), while the second is calculating the versiondiff by radix.

Counter versiondiff: For this versiondiff schema, differences between minor versions and differences between major versions are treated similarly. For example, if one software has three versions: 1.1, 1.2, 2.0, then the versions will be assigned counters 1, 2, 3 based on the rank of their values. Therefore, the versiondiff between 1.1 and 1.2 is the same as the one between 1.2 and 2.0.

Radix-based versiondiff: Intuitively, the difference between major versions is more significant than the difference between minor versions. Thus, when calculating versiondiff, we need to assign a higher weight to relatively major version changes and lower weight to relatively minor version changes. For example, for the three versions 1.0, 1.1, 2.0, if we assign a weight of 10 to the major version and a weight of 1 to each minor version, the versiondiff between 1.1 and 1.0 will be 1, while the versiondiff between 2.0 and 1.1 will be 9.

When analyzing the data, we found out that versiondiff did not work very well for our problem because, in most cases, the new vulnerabilities affect all previous versions as well. Therefore, most values of versiondiff are zero, as the new vulnerability instance must affect an older version that also exists in the previous instance, thus, resulting in a versiondiff of zero. In order to mitigate this limitation, we created another predictive feature for our later experiments. The additional feature that we constructed is the number of occurrences of a certain version of each software. More details will be provided in Section 4.

3.3 Machine Learning Functions

We used either classification or regression functions for our prediction, depending on how we define the predicted feature. The TTNV could be a number representing how many days we need to wait until the occurrence of the next vulnerability. Or it could be binned and each bin stands for values within a range. For the former case, we used regression functions. For the latter case, we used classification functions. We used WEKA [5] implementations of machine learning algorithms to build predictive models for our data. For both regression and classification cases, we explored all of the functions compatible to our data type, with default parameters. In the case of the regression problem, the compatible functions are: linear regression, least median square, multi-layer perceptron, RBF network, SMO regression, and Gaussian processes. In the case of classification, the compatible functions are: logistic, least median square, multi-layer perceptron, RBF network, SMO, and simple logistic.

4 Experimental Results

We conducted the experiments on our department's computer cluster - Beocat. We used a single node and 4G RAM for each experiment. As mentioned above, WEKA [5], a data-mining suite, was used in all experiments.

4.1 Evaluation Metrics

A number of metrics are used to evaluate the performance of the predictive models learned.

Correlation Coefficient: The correlation coefficient is a measure of how well trends in the predicted values follow trends in actual values. It is a measure of how well the predicted values from a forecast model "fit" the real-life data. The

correlation coefficient is a number between -1 and 1. If there is no relationship between the predicted values and the actual values, the correlation coefficient is close to 0 (i.e., the predicted values are no better than random numbers). As the strength of the relationship between the predicted values and actual values increases, so does the correlation coefficient. A perfect fit gives a coefficient of 1.0. Opposite but correlated trends result in a correlation coefficient value close to -1. Thus, the higher the absolute value of the correlation coefficient, the better; however, when learning a predictive model, negative correlation values are not usually expected. We generate correlation coefficient values as part of the evaluation of the regression algorithms used in our study.

Root Mean Squared Error: The mean squared error (MSE) of a predictive regression model is another way to quantify the difference between a set of predicted values, x_p, and the set of actual (target) values, x_t, of the attributed being predicted. The root mean squared error (RMSE) can be defined as:

$$\text{RMSE}(x_p,\, x_t) = \sqrt{MSE(x_p, x_t)} = \sqrt{E[(x_p - x_t)^2]} = \sqrt{\frac{\sum_{i=1}^{n} (x_{p,i} - x_{t,i})^2}{n}}$$

Root Relative Squared Error: According to [1], the root relative squared error (RRSE) is relative to what the error would have been if a simple predictor had been used. The simple predictor is considered to be the mean/majority of the actual values. Thus, the relative squared error takes the total squared error and normalizes it by dividing by the total squared error of the simple predictor. By taking the root of the relative squared error one reduces the error to the same dimensions as the quantity being predicted.

$$\text{RRSE}(x_p,\, x_t) = \sqrt{\frac{\sum_{i=1}^{n} (x_{p,i} - x_{t,i})^2}{\sum_{i=1}^{n} (x_{t,i} - \bar{x})^2}}$$

Correctly Classified Rate: To evaluate the classification algorithms investigated in this work, we use a metric called correctly classified rate. This metric is obtained by dividing correctly classified instances by all instances. Obviously, a higher value suggests a more accurate classification model.

4.2 Experiments

We performed a large number of experiments, by using different versiondiff schemes, different time schemes, and by including CVSS metrics or not. For different software, different feature combinations produce the best results. Hence, we believe it is not effective to build a single model for all the software. Instead, we build separate models for different software. This way, we also avoid potential

scalability issues due to the large number of nominal type values from vendor names and software names.

Given the large number of vendors in the data, we did not run experiments for all of them. We focused especially on several major vendors (Linux, Microsoft, Mozilla and Google) and built vendor-specific models. For three vendors (Linux, Microsoft and Mozilla), we also built software-specific models. For other vendors (Apple, Sun and Cisco), we bundled all their software in one experiment.

4.3 Results

Linux: We used two versiondiff schemes, specifically counter-based and radix-based, to find out which one is more effective for our model construction. We also compared two different time schemes (epoch time, and using month and day separately). In a first set of experiments, we predicted TTNV based on regression models. In a second set of experiments, we grouped the predictive feature (TTNV) values into bins, as we observed that the TTNV distribution shows several distinct clusters, and solved a classification problem.

Table 1 shows the results obtained using the *epoch time* scheme versus the results obtained using the *month and day* scheme, in terms of correlation coefficient, for regression models. As can be seen, the results of our experiments did not show a significant difference between the two time schemes that we used, although we expected the month and day feature to provide better results than the absolute epoch time, as explained in Section 3.2. Thus, neither scheme has acceptable correlation capability on the test data. We adapted the month and day time schema for all of the following experiments.

Table 2 shows a comparison of the results of the two different versiondiff schemes. As can be seen, both perform poorly as well. Given the unsatisfactory results, we believed that the large number of Linux sub-versions could be potentially a problem. Thus, we also investigated constructing the versiondiff feature by binning versions of the Linux kernel (to obtained a smaller set of sub-versions). We round each sub-version to its third significant major version (*e.g.* Bin(2.6.3.1) = 2.6.3). We bin based on the first three most significant versions because more than half of the instances (31834 out of 56925) have version longer than 3, and Only 1% (665 out of 56925) versions are longer than 4. Also, the

Table 1. Correlation Coefficient for Linux Vulnerability Regression Models Using Two Time Schemes

Regression Functions	Epoch time		Month and day	
	training	test	training	test
Linear regression	0.3104	0.1741	0.6167	-0.0242
Least mean square	0.1002	0.1154	0.1718	0.1768
Multi-layer perceptron	0.2943	0.1995	0.584	-0.015
RBF network	0.2428	0	0.1347	0.181
SMO regression	0.2991	0.2186	0.2838	0.0347
Gaussian processes	0.3768	-0.0201	0.8168	0.0791

Table 2. Correlation Coefficient for Linux Vulnerability Regression Models Using Two Versiondiff Schemes

Regression Functions	Version counter		Radix based	
	training	test	training	test
Linear regression	0.6167	-0.0242	0.6113	0.0414
Least mean square	0.1718	0.1768	0.4977	-0.0223
Multi-layer perceptron	0.584	-0.015	0.6162	0.1922
RBF network	0.1347	0.181	0.23	0.0394
SMO regression	0.2838	0.0347	0.2861	0.034
Gaussian processes	0.8168	0.0791	0.6341	0.1435

Table 3. Correlation Coefficient for Linux Vulnerability Regression Models Using Binned Versions versus Non-Binned Versions

Regression Functions	Non-binned versions		Binned versions	
	training	test	training	test
Linear regression	0.6113	0.0414	0.6111	0.0471
Least mean square	0.4977	-0.0223	0.5149	0.0103
Multi-layer perceptron	0.6162	0.1922	0.615	0.0334
RBF network	0.23	0.0394	0.0077	-0.0063
SMO regression	0.2861	0.034	0.285	0.0301
Gaussian processes	0.6341	0.1435	0.6204	0.1369

difference on the third subversion will be regarded as a huge dissimilarity for Linux kernels. We should note that the sub-version problem may not exist for other vendors, such as Microsoft, where the versions of the software are naturally discrete (all Microsoft products have versions less than 20). Table 3 shows the comparisons between regression models that use binned versions versus regression models that do not use binned versions. The results are still not good enough as many of the versiondiff values are zero, as explained in Section 3.2 (new vulnerabilities affect affect previous versions as well).

TTNV Binning: Since we found that the feature (TTNV) of Linux shows distinct clusters, we divided the feature values into two categories, more than 10 days and no more than 10 days, thus transforming the original regression problem into an easier binary classification problem. The resulting models are evaluated in terms of corrected classified rates, shown in Table 4. While the models are better in this case, the false positive rates are still high (typically above 0.4). In this case, as before, we used default parameters for all classification functions. However, for the SMO function, we also used the Gaussian (RBF) kernel. The results of the SMO (RBF kernel) classifier are better than the results of most other classifiers, in terms of correctly classified rate. However, even this model has a false positive rate of 0.436, which is far from acceptable.

CVSS Metrics: In all cases, we also perform experiments by adding CVSS metrics as predictive features. However, we did not see much differences.

Table 4. Correctly Classified Rates for Linux Vulnerability Classification Models Using Binned TTNV

Classification Functions	Correctly classified		FPR	TPR
	training	test		
Simple logistic	97.6101%	69.6121%	0.372	0.709
Logistic regression	97.9856%	57.9542%	0.777	0.647
Multi-layer perceptron	98.13%	64.88%	0.689	0.712
RBF network	95.083%	55.18%	0.76	0.61
SMO	97.9061%	61.8259%	0.595	0.658
SMO (RBF kernel)	96.8303%	62.8392%	0.436	0.641

Microsoft: As we have already observed the limitation of versiondiff scheme in the analysis of Linux vulnerabilities, for Microsoft instances, we use only the number of occurrences of a certain version of a software or occurrences of a certain software, instead of using versiondiff, as described below. We analyzed the set of instances and found out that more than half of the instances do not have version information. Most of these case are Windows instances. Most of the non-Windows instances (more than 70%) have version information. Therefore, we used two different occurrence features for these two different types of instances. For Windows instances, we used the occurrence of each software as a predictive feature. For non-Windows instances, we used the occurrence of each version of the software as a predictive feature.

Also based on our observations for Linux, we used only the month and day scheme, and did not use the epoch time scheme in the set of experiments we performed for Windows. We analyzed instances to identify potential clusters of TTNV values. However, we did not find any obvious clusters for either windows or non-windows instances. Therefore, we only used regression functions. The results obtained using the aforementioned features for both Windows and non-Windows instances are presented in Table 5. As can be seen, the correlation coefficients are still less than 0.4.

We further investigated the effect of building models for individual non-Windows applications. For example, we extracted Internet Explorer (IE) instances and build several models for this set. When CVSS metrics are included,

Table 5. Correlation Coefficient for Windows and Non-Windows Vulnerability Regression Models, Using Occurrence Version/Software Features and Day and Month Time Scheme

Regression Functions	Win Instances		Non-win Instances	
	training	test	training	test
Linear regression	0.4609	0.1535	0.5561	0.0323
Least mean square	0.227	0.3041	0.2396	0.1706
Multi-layer perceptron	0.7473	0.0535	0.5866	0.0965
RBF network	0.1644	0.1794	0.1302	-0.2028
SMO regression	0.378	0.0998	0.4013	-0.0467
Gaussian processes	0.7032	-0.0344	0.7313	-0.0567

the correlation coefficient is approximately 0.7. This is better than when CVSS metrics are not included, in which case, the correlation coefficient is approximately 0.3. The results showing the comparison between IE models with and without CVSS metrics is shown in Table 6. We tried to performed a similar experiment for Office. However, there are only 300 instances for Office. Other office-related instances are about individual software such as Word, PowerPoint, Excel and Access, *etc*, and each has less than 300 instances. Given the small number of instances, we could not build models for Office.

Table 6. Correlation Coefficient for IE Vulnerability Regression Models, with and without CVSS Metrics

Regression Functions	With CVSS		Without CVSS	
	training	test	training	test
Linear regression	0.8023	0.6717	0.7018	0.3892
Least mean square	0.6054	0.6968	0.4044	0.0473
Multi-layer perceptron	0.9929	0.6366	0.9518	0.0933
RBF network	0.1381	0.0118	0.151	-0.1116
SMO regression	0.7332	0.5876	0.5673	0.4813
Gaussian processes	0.9803	0.6048	0.9352	0.0851

Mozilla: At last, we built classification models for Firefox, with and without the CVSS metrics. The results are shown in Table 7. As can be seen, the correctly classified rates are relatively good (approximately 0.7) in both cases. However, the number of instances in this dataset is rather small (less than 5000), therefore it is unclear how stable the prediction model is.

Table 7. Correctly Classified Rate for Firefox Vulnerability Models with and without CVSS Metrics

Classification Functions	With CVSS		Without CVSS	
	training	test	training	test
Simple logistic	97.5%	71.4%	97.5%	71.4%
Logistic regression	97.5%	70%	97.8%	70.5%
Multi-layer perceptron	99.5%	68.4%	99.4%	68.3%
RBF network	94.3%	71.9%	93.9%	67.1%
SMO	97.9%	55.3%	97.4%	55.3%

4.4 Parameter Tuning

As mentioned above, we used default parameters for all regression and classification models that we built. To investigate if different parameter settings could produce better results, we chose to tune parameters for the support vector machines algorithm (SVM), whose WEKA implementations for classifications and regression are called SMO and SMO regression, respectively. There are two main

parameters that can be tuned for SVM, denoted by C and σ. The C parameter is a cost parameter which controls the trade-off between model complexity and training error, while σ controls the width of the Gaussian kernel [2].

To find the best combination of values for C and σ, we generated a grid consisting of the following values for C: 0.5, 1.0, 2.0, 3.0, 5.0, 7.0, 10, 15, 20 and the following values for σ: 0, 0.05, 0.1, 0.2, 0.3, 0.5, 1.0, 2.0, 5.0, and run the SVM algorithm for all possible combinations. We used a separate validation set to select the combination of values that gives the best values for correlation coefficient, and root squared mean error and root relative squared error together. The validation and test datasets have approximately equal sizes; the test set consists of chronologically newer data, as compared to the validation data, while the validation data is newer than the training data.

Table 8 shows the best parameter values when tuning was performed based on the correlation coefficient, together with results corresponding to these parameter values, in terms of correlation coefficient, RRSE and RMSE (for both validation and test datasets). Table 9 shows similar results when parameters are tuned on RRSE and RMSE together.

Table 8. Parameter Tuning Based on Correlation Coefficient

Group Targeted	Parameters		Validation			Test		
	C	G	RMSE	RRSE	CC	RMSE	RRSE	CC
Adobe CVSS	3.0	2.0	75.2347	329.2137%	0.7399	82.2344	187.6%	0.4161
IE CVSS	1.0	1.0	8.4737	74.8534%	0.4516	11.6035	92.2%	-0.3396
Non-Windows	1.0	0.05	92.3105	101.0356%	0.1897	123.4387	100.7%	0.223
Linux CVSS	15.0	0.1	12.6302	130.8731%	0.1933	45.0535	378.3%	0.2992
Adobe	0.5	0.05	43.007	188.1909%	0.5274	78.2092	178.5%	0.1664
IE	7.0	0.05	13.8438	122.2905%	0.2824	14.5263	115.5%	-0.0898
Apple Separate	3.0	0.05	73.9528	104.0767%	0.2009	91.1742	116.4%	-0.4736
Apple Single	0.5	0.0	493.6879	694.7868%	0	521.228	1401.6%	0
Linux Separate	2.0	0.05	16.2225	188.6665%	0.3105	49.8645	418.7%	-0.111
Linux Single	1.0	0.05	11.3774	83.2248%	0.5465	9.4743	79.6%	0.3084
Linux Binned	2.0	0.05	16.2225	188.6665%	0	49.8645	418.7%	-0.111
Windows	5	0.05	21.0706	97.4323%	0.1974	72.1904	103.1%	0.1135

4.5 Summary

The experiments above indicate that it is hard to build good prediction models based on the limited data available in NVD. For example, there is no version information for most Microsoft instances (especially, Windows instances). Some results look promising (*e.g.* the models we built for Firefox), but they are far from usable in practice. Below, we discuss what we believe to be the main reasons for the difficulty of building good prediction models for TTNV from NVD.

Table 9. Parameter Tuning Based on RMSE and RRSE

Group Targeted	Parameters		Validation			Test		
	C	G	RMSE	RRSE	CC	RMSE	RRSE	CC
Adobe CVSS	0.5	0.2	19.4018	84.8989%	0.2083	61.2009	139.6667%	0.5236
IE CVSS	2.0	1.0	8.4729	74.8645%	0.4466	11.4604	91.1018%	-0.3329
Non-Windows	0.5	0.1	91.1683	99.7855%	0.188	123.5291	100.7%	0.2117
Linux CVSS	2.0	0.5	7.83	81.1399%	0.1087	19.1453	160.8%	0.3002
Adobe	1.0	0.5	19.5024	85.3392%	-0.4387	106.2898	242.5%	0.547
IE	0.5	0.3	12.4578	110.0474%	0.2169	13.5771	107.9%	-0.1126
Apple Separate	7.0	1.0	70.7617	99.5857%	0.1325	80.2045	102.4%	-0.0406
Apple Single	0.5	0.05	75.9574	106.8979%	-0.3533	82.649	105.5%	-0.4429
Linux Separate	0.5	2.0	14.5428	106.3799%	0.2326	18.5708	155.9%	0.1236
Linux Single	5.0	0.5	10.7041	78.2999%	0.4752	12.3339	103.6%	0.3259
Linux Binned	0.5	2.0	14.5428	106.3799%	0.2326	18.5708	155.9%	0.1236
Windows	5.0	0.05	21.0706	97.4323%	0.1974	72.1904	103%	0.1135

4.6 Discussion

We believe the main factor affecting the predictive power of our models is the low quality of the data from the National Vulnerability Database. Following are several limitations of the data:

- Missing information: most instances of Microsoft do not have the version information, without which we could not observe how the number of vulnerabilities evolves over versions.
- "Zero" versiondiffs: most of versiondiff values are zero because earlier-version applications are also affected by the later-found vulnerabilities (this is assumed by a number of large companies, *e.g.* Microsoft and Adobe) and significantly reduces the utility of this feature.
- Vulnerability release time: The release date of vulnerability could largely be affected by vendors' practices. For example, Microsoft usually releases their vulnerability and patch information on the second Tuesday of each month, which may not accurately reflect the discovery date of the vulnerabilities.
- Data error: We found a number of obvious errors in NVD, such as the aforementioned Linux kernel version error.

5 Related Works

Alhazmi and Malaiya [3] have addressed the problem of building models for predicting the number of vulnerabilities that will appear in the future. They targeted operating systems instead of applications. The Alhazmi-Malaiya Logistic model works well for fitting existing data, when evaluated in terms of average error (AE) and average bias (AB) of number of vulnerabilities over time. However, fitting existing data is a prerequisite of testing models: predictive power is

the most important criteria [18] . They did test the predictive accuracy of their models and got satisfactory results [18].

Ozment [19] examined the vulnerability discovery models (proposed by Alhazmi Malaiya [3]) and pointed some limitations that make these models inapplicable. One of them is that there is not enough information included in a government supported vulnerability database (*e.g.* National Vulnerability Database). This is confirmed by our empirical study.

McQueen *et al.* [15] designed algorithms for estimating the number of zero-day vulnerabilities on each given day. This number can indicate the overall risk level from zero-day vulnerabilities. However, for different applications the risks could be different. Our work aimed to construct software-specific prediction models.

Massacci *et al.* [14,16] compared several existing vulnerability databases based on the type of vulnerability features available in each of them. They mentioned that many important features are not included in most databases. *e.g.* discovery date is hard to find. Even though certain databases (such as OSVDB that as we also studied) claim they include the features, most of the entries are blank. For their Firefox vulnerability database, they employed textual retrieval techniques and took keywords from CVS developer's commit log to get several other features by cross-referencing through CVE ids. They showed that by using two different data sources for doing the same experiment, the results could be quite different due to the high degree of inconsistency in the data available for the research community at the current time. They further tried to confirm the correctness of their database by comparing data from different sources. They used data-mining techniques (based on the database they built) to prioritize the security level of software components for Firefox.

Ingols *et al.* [10] tried to model network attacks and countermeasures using attack graphs. They pointed out the dangers from zero-day attacks and also mentioned the importance of modeling them. There has been a long line of attack-graph works [4,7,8,9,11,12,13,17,20,22] which can potentially benefit from the estimation of the likelihood of zero-day vulnerabilities in specific applications.

6 Conclusions

In this paper we present our effort in building prediction models for zero-day vulnerabilities based on the information contained in the National Vulnerability Database. Our research found that due to a number of limitations of this data source, it is unlikely that one can build a practically usable prediction model at this time. We presented our rigorous evaluation of various feature construction schemes and parameter tuning for learning algorithms, and notice that none of the results obtained shows acceptable performance. We discussed possible reasons as of why the data source may not be well suited to predict the desired features for zero-day vulnerabilities.

References

1. Root relative squared error. Website, http://www.gepsoft.com/gxpt4kb/Chapter10/Section1/SS07.htm
2. Support vector machines. Website, http://www.dtreg.com/svm.htm
3. Alhazmi, O.H., Malaiya, Y.K.: Prediction capabilities of vulnerability discovery models. In: Annual Reliability and Maintainability Symposium, RAMS (2006)
4. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: 9th ACM Conference on Computer and Communications Security, CCS (2002)
5. Bouckaert, R.R., Frank, E., Hall, M., Kirkby, R., Reutemann, P., Seewald, A., Scuse, D.: WEKA Manual for Version 3.7. The University of Waikato (2010)
6. Buttner, A., Ziring, N.: Common platform enumeration (cpe) c specification. Technical report, The MITRE Corporation AND National Security Agency (2009)
7. Dacier, M., Deswarte, Y., Kaâniche, M.: Models and tools for quantitative assessment of operational security. In: IFIP SEC (1996)
8. Dawkins, J., Hale, J.: A systematic approach to multi-stage network attack analysis. In: Proceedings of Second IEEE International Information Assurance Workshop, pp. 48–56 (April 2004)
9. Dewri, R., Poolsappasit, N., Ray, I., Whitley, D.: Optimal security hardening using multi-objective optimization on attack tree models of networks. In: 14th ACM Conference on Computer and Communications Security, CCS (2007)
10. Ingols, K., Chu, M., Lippmann, R., Webster, S., Boyer, S.: Modeling modern network attacks and countermeasures using attack graphs. In: 25th Annual Computer Security Applications Conference, ACSAC (2009)
11. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: 22nd Annual Computer Security Applications Conference (ACSAC), Miami Beach, Florida (December 2006)
12. Jajodia, S., Noel, S., O'Berry, B.: Topological analysis of network attack vulnerability. In: Kumar, V., Srivastava, J., Lazarevic, A. (eds.) Managing Cyber Threats: Issues, Approaches and Challanges, ch. 5. Kluwer Academic Publisher, Dordrecht (2003)
13. Lippmann, R., Ingols, K.W.: An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory (March 2005)
14. Massacci, F., Nguyen, V.H.: Which is the right source for vulnerability studies? an empirical analysis on mozilla firefox. In: MetriSec (2010)
15. McQueen, M., McQueen, T., Boyer, W., Chaffin, M.: Empirical estimates and observations of 0day vulnerabilities. In: 42nd Hawaii International Conference on System Sciences (2009)
16. Nguyen, V.H., Tran, L.M.S.: Predicting vulnerable software components with dependency graphs. In: MetriSec (2010)
17. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: 13th ACM Conference on Computer and Communications Security (CCS), pp. 336–345 (2006)
18. Ozment, A.: Improving vulnerability discovery models analyzer. In: QoP 2007 (2007)
19. Ozment, A.: Vulnerability Discovery & Software Security. PhD thesis, University of Cambridge (2007)

20. Phillips, C., Swiler, L.P.: A graph-based system for network-vulnerability analysis. In: NSPW 1998: Proceedings of the 1998 Workshop on New Security Paradigms, pp. 71–79. ACM Press, New York (1998)
21. Schiffman, M., Eschelbeck, G., Ahmad, D., Wright, A., Romanosky, S.: CVSS: A Common Vulnerability Scoring System. National Infrastructure Advisory Council (NIAC) (2004)
22. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 254–265 (2002)

Similar Subsequence Search in Time Series Databases

Shrikant Kashyap, Mong Li Lee, and Wynne Hsu

School of Computing
National University of Singapore
Singapore
{shrikant,leeml,whsu}@nus.edu.sg

Abstract. Finding matching subsequences in time series data is an important problem. The classical approach to search for matching subsequences has been on the principle of exhaustive search, where all possible candidates are generated and evaluated or all the terms of the time series in a data base are examined. As a result most of the subsequence search algorithms are cubic in nature with few algorithms of quadratic nature. Some approximate algorithms have been proposed, as a result, to speed up the search for matching subsequences. In this work, we propose a fast and efficient exact subsequence search algorithm which is subquadratic in nature. We introduce the notion of *eHaar (envelope Haar)* to prune parts of the time series data which will not contain subsequences that can match the query subsequence. This pruning phase dramatically reduces the search space, thus allowing dynamic time warping based subsequence search techniques to be applied on gigabyte-size time series databases. Experiment results demonstrate that the proposed approach outperforms existing state-of-the art techniques.

1 Introduction

Advancement of technology has led to the huge repositories of time series data, whose sizes range from gigabytes to terabytes and beyond. Various application domains generate time series data such as financial data, RFID data, sensor data, music data, etc. One fundamental task in time series database is the search for similar subsequences. Given a query sequence and a database of time series, the aim is to search for subsequences which are most similar to the query sequence. Efficient subsequence search is crucial for datasets where the time series data tend to be extremely long compared to the query sequence and the number of time series in the database is large.

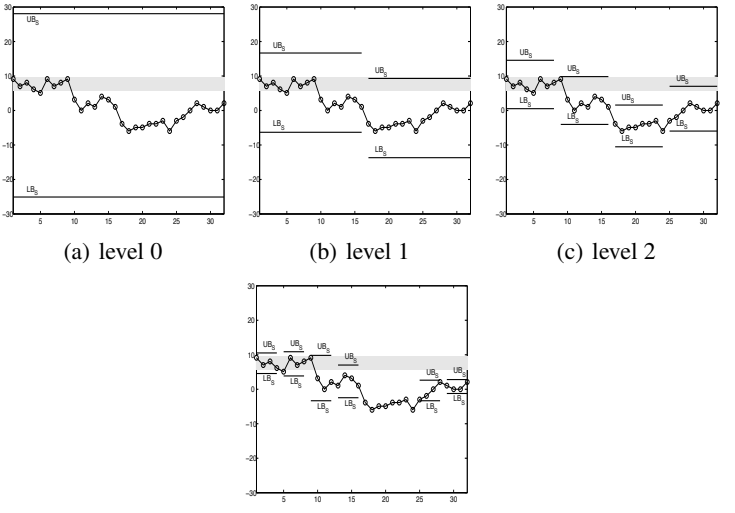
Many DTW (Dynamic Time Warping) based algorithms have been proposed for subsequence matching. The advantage of DTW based algorithms over Euclidean measures is that they are robust to misalignments and time warps. However, these algorithms are computationally intensive and have a complexity proportional to $O(l * l * N)$ where l is the length of a query sequence and N is the length of the time series. The current state-of-the art is the SPRING method proposed in [15] where the time complexity is reduced to $O(l * N)$. In spite of this improvement, subsequence matching remains expensive, especially on gigabyte size databases.

In this work, we propose a fast subsequence matching approach that utilizes the notion of an *envelope Haar (eHaar)* to prune off portions of time series that cannot

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 232–246, 2011.

contain matching subsequences. *Envelope Haar* is based on the Haar wavelet's theory of energy conservation. Together with the hierarchy of Haar wavelet coefficients, we can create envelopes around parts of a time series. These envelopes establish the upper and lower bounds for various parts of the time series.

Figure 1(a) shows a time series and its upper and lower bounds computed based on the Haar wavelet coefficients at level 0. The bounds for the query sequence is indicated by the grey band. Figures 1(b)-(d) shows the tightening of bounds for each successive level of Haar wavelet coefficients. We observe that at level 2 (Figure 1(c)), the bounds of the time series from time points 17 to 24 clearly do not intersect with the bounds of the query sequence. Hence, we can safely prune off this portion of the time series. At level 3 (see Figure 1(d)), more parts of the time series (time points 25 to 32) are pruned. The remaining portions of the time series are the candidates for subsequence matching. Experiment results on a 5 GB time series dataset show that 64% of a time series remains after pruning. With this reduced set of time series, it is now feasible to apply any existing DTW-based algorithms to obtain the final set of matching subsequences.



(a) level 0 (b) level 1 (c) level 2

(d) level 3

Fig. 1. Example to illustrate the use of bounds to prune of parts of time series

The contributions of this work are as follows:

1. We propose a method that provides a fast pruning of the search space and limits the computationally expensive DTW calculations to a selected set of candidates.
2. Using the properties of Haar wavelets to create envelopes, the accuracy of subsequence search is not sacrificed compared to approximate methods.
3. The method adapts well to time series databases of large size, with experiments being reported on gigabyte-size databases.

In section 2 we recall the concept of Haar wavelets and the unique properties it has as a multi-resolution representation technique. Using these techniques, we develop

the concept of envelope Haar (eHaar) in section 4. Further in section 6 we describe our main algorithm and the pruning efficiency that eHaar brings along with it. Section 7 explains the experimental results and comparisons with the classical methods. We include a review of related works in the area of subsequence matching in section 8 and we conclude with the section 9.

2 Preliminaries

In this section we describe the Haar wavelet transform which forms the basis of our envelope Haar concept. Haar wavelet is the simplest form of wavelet. It works by calculating the sums and differences of adjacent elements.

Given a set of discrete signals [9 7 5 3 0 2 -4 -6], the Haar wavelet transform results in [(8 1) (4 1) (1 -1) (-5 -1)] at level 1. The process is repeated on the average values [8 4 1 -5] to obtain the level 2 decomposition [(6 2) (-2 3)], and so on. The full decomposition is shown in Table 1. Note that no information is gained or lost by this process.

Table 1. Wavelet coefficients for the time series 9 7 5 3 0 2 -4 -6

Resolution	Averages(A_i)	Detail Coefficients(H_i)
8	[9 7 5 3 0 2 -4 -6]	
4	[8 4 1 -5]	[1 1 -1 -1]
2	[6 -2]	[2 3]
1	[2]	[4]

2.1 Conservation of Energy

An important property of the Haar transform is that it conserves the energy of the signals. Given a signal S with N values, the energy of the signal S is the sum of the squares of its values. That is, the energy of the signal S, E_S is defined by

$$E_S = S_1^2 + S_2^2 + \ldots + S_N^2 \tag{1}$$

Consider the example in Table 1. We have

$$E = 2^3 * (2^2 + 4^2) + 2^2 * (2^2 + 3^2) + 2 * (1^2 + 1^2 + (-1)^2 + (-1)^2)) = 220.$$

3 Storage Scheme

Before we describe the nature of eHaar, we would like to describe the storage scheme used for the coefficients. This will help to understand the concept of eHaar and access to the coefficients in a vertical fashion (levelwise).

eHaar representation warrants storing of the N coefficients along with the total energy E of the time series for each time series. Therefore, $N + 1$ terms are stored for each time series. Note that the original time series can be regained from N coefficients, hence the original time series data can be discarded.

eHaar representation requires that the data must be accessed in a vertical fashion. In the new storage system, coefficients belonging to one level are stored together, with different levels being stored in different files. Classically data belonging to one time series is stored together.

Thus the coefficients are stored level-by-level: level 0 for all time series, level 1 for all time series and so on. Also the energy terms are stored together. As we need to store data efficiently in the disk, the "higher" level coefficients of eHaar of more than one time series can be stored per disk page. However for "lower" level coefficients which have a high number of coefficients per time series need to be stored in more than one disk page.

In more detail, level ℓ of the Haar tree (Figure 2) includes 2^ℓ coefficients (level 0 has 2 terms). Assuming a page size of B bytes and b bytes per term, a page stores $\frac{B}{b}$ terms. Thus, level $\log \frac{B}{b}$ of a single record fits exactly in one disk page. Thereafter, level ℓ of each record occupies $2^{\ell - \log \frac{B}{b}}$ disk pages. For $\ell < \log \frac{B}{b}$, we store the 2^ℓ level-ℓ coefficients of $2^{\log \frac{B}{b} - \ell}$ time series on the same disk page. For example, if $B = 1$KB and $b = 4$bytes, then the 128 level-7 coefficients of 2 records are stored on the same page.

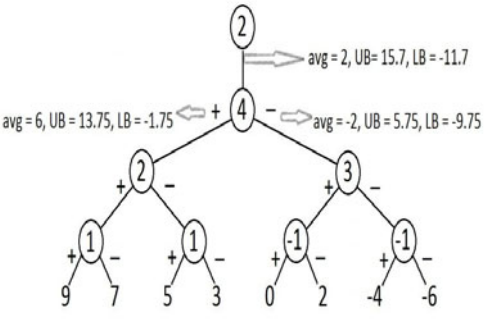


Fig. 2. eHaar coefficients and bounds for the first two levels

4 Envelope Haar

The property of energy conservation in the Haar wavelet transform enables us to develop bounds on the time series data. Given a time series data $S = S_1, S_2, S_3, ..., S_{n-1}, S_n$, we can define the set of Discrete Haar Wavelet Transform (DHWT) coefficients as $C = C_1, C_2, ..., C_n$. The coefficient C_1 is actually the overall average value of the time series S, that is, $C_1 = A_1$. Further, we have the coefficients $C_2 = H_1, C_3 = H_2, \ldots C_n = H_{n-1}$. Note that now the time series S can be denoted using the coefficients only $S = C$ as S can be fully recovered using the coefficient of C. Hence, $S = C_1, C_2, ..., C_n$.

For our running example in Table 1, the coefficients are $C_1 = 2, C_2 = 4, C_3 = 2, C_4 = 3, C_5 = 1, C_6 = 1, C_7 = -1, C_8 = -1$. Apart from the coefficients we also store E_S, the total energy of the time series. Note that

$$E_S = 2^{log_2 N}(C_1^2 + C_2^2) + 2^{log_2 N - 1}(C_3^2 + C_4^2) + \ldots + 2^1(\ldots + C_{N-1}^2 + C_N^2)$$

The tightest possible upper and lower bounds for a time series would be

$$avg_S \pm max(S_i - avg_S)$$

where avg_S is the average value of the signal S. Note that avg_S is equal to C_1. We have

$$max(S_i - avg_S)^2 \leq \sum_i (S_i - avg_S)^2 \tag{2}$$

$$\begin{aligned}\sum_i (S_i - avgS)^2 &= \sum_i S_i^2 + \sum_i avg_S^2 - \sum_i 2 * S_i * avg_S \\ &= E_S + N * avg_S^2 - 2 * avg_S^2 * N \\ &= E_S - N * avg_S^2 \\ &= E_S - N * C_1^2\end{aligned}$$

Hence the upper bound UB and lower bound LB for the time series S is given by

$$UB = C_1 + \sqrt{E_S - N * C_1^2} \tag{3}$$

and

$$LB = C_1 - \sqrt{E_S - N * C_1^2} \tag{4}$$

Since Haar is a multi-resolution transform, the wavelet coefficients up to a particular level can be read and along with suitable energy term, bounds can be defined for that particular level. When the second level of coefficients is read, the bounds for one of the two halves of the time series would be

$$UB_1 = C_1 + C_2 + \sqrt{E_S - N * C_1^2 - N * C_2^2} \tag{5}$$

$$LB_1 = C_1 + C_2 - \sqrt{E_S - N * C_1^2 - N * C_2^2} \tag{6}$$

A general formula for the bounds can be derived hence. In the equations L refers to the level of operation and i is the sequence number for that particular level.

$$UB_{Li} = avg_S + R, LB_{Li} = avg_S - R$$

where $avg_S = C_1 - \sum_{j=1}^{L}(-1)^{\lceil i/2^{j-1} \rceil} * C_{\lceil (2^L + i)/2^j \rceil}$ and $R = \sqrt{E_S - N * C_1^2 - \sum_{j=L-1}^{2^L - 1} \sum_{k=1}^{j} 2^{log_2 N - k} * C_{1+k}^2}$. The initial value of R will be $\sqrt{(E_S - N * C_1^2)}$ when only the first level of coefficients has been read. Reading of the next level of coefficients, the value of R would be updated to $\sqrt{E_S - N * C_1^2 - N * C_2^2}$.

Figure 2 shows the coefficients and the corresponding bounds for the first two levels of the example time series. Note that as we read additional levels of coefficients, tighter bounds get generated. When all the coefficients are read, both the upper and lower bounds converge to the time series.

5 Reading the Coefficients

As we have already discussed that coefficients at one level are stored together. In the light of the fact that eHaar accesses the coefficients levelwise, it is important to note that the order of coefficients is preserved while storing them so that coefficients are used in an order. For example, if at an instance when we have the time series divided into 2 parts, we have to read 2 coefficients at the next level to further divide the time series. The coefficients will have to be read in-order and used for the 2 parts in an orderly fashion (1 for each part) to generate the subsequent 4 parts of the time series. Thereafter, when the next level of coefficients is accessed, it has 4 coefficients in all. These 4 coefficients will be accessed again in-order and assigned to the 4 parts generated in-order to further derive the 8 parts of the time series.

Referring to the figure 2 for the time series $9, 7, 5, 3, 0, 2, -4, -6$, when the first level coefficient 2 is read, we have one set of bounds for the whole time series. After reading the coefficient 4 at the next level (2^{nd}), we have two sets of bounds for the two parts of the time series $S1(9, 7, 3, 5)$ and $S2(0, 2, -4, -6)$. Now for the first part $S1$,we need to read only the first coefficient 2 at the 3^{rd} level while for the second part $S2$, we need to read only the second coefficient 3 at the 3^{rd} level. This order is maintained while storing coefficients. After reading the coefficients at the 3^{rd} level, the time series can be divided into 4 parts, $S1(9, 7), S2(5, 3), S3(0, 2), S4(-4, -6)$. Note that for each part we need to read only 1 coefficient from the 4^{th} level for each part.

Concluding the section, the main aim of the discussion is to show that at any stage of recursion we need to read one specific coefficient for a specific part of the time series and this is possible if we store and retrieve coefficients in order. At the same time we should not forget that these parts of time series are being obtained from the coefficients in a hierarchical manner and the original terms have already been discarded during the coefficient generation phase.

6 Subsequence Matching Using eHaar

In this section we describe the proposed subsequence search using eHaar. The first step utilizes the bounds to prune parts of the time series that do not contain matches to the query subsequence. The next step employs any existing subsequence search methods to retrieve the results in the remaining time series. In this work, we use the SPRING method [15] as this is currently the most efficient method for DTW based matching of sequences.

6.1 Pruning

Pruning of time series data for subsequence search is one area which has not been explored well. EBSM does prune data at an alarming rate, but the problem with the method is that it tends to be an approximate method. To the best of our knowledge, no other subsequence search method prunes data effectively to reduce the size of candidate list.

In this step, we take advantage of the bounds created by eHaar around the time series and the query subsequence to prune data effectively. The basic premise is that if the bounds of the subsequence intersect the bounds of the time series or intersect the parts of the time series, that time series or a part of time series is likely to have a subsequence matching the query subsequence. However, if the bounds of the time series and the subsequences do not intersect, the chance of discovering a matching subsequence is zero.

The advantage that eHaar brings with it is that it can not only give a lower bound, it can also create an upper bound for a time series and the query subsequence. Also the multi-resolution nature of eHaar assures that bounds are tightened as we read more and more coefficients, guaranteeing pruning of candidates as early as possible. It should be noted that as we read more and more coefficients, we develop bounds for not just the time series, but of the parts of the time series as well. This developing of bounds for the parts helps us when the bounds of the time series intersect the bounds of the query subsequence. We can prune the parts of the time series whose bounds do not intersect the bounds of the query subsequence, with the result being that we need to examine only parts of time series for subsequence matching. (Note that in case there are two consecutive parts of the time series whose bounds intersect those of the query subsequence, we merge them in the final step for the DTW based subsequence search). Algorithm 1 describes the pruning part of the method proposed while algorithm 2 describes the overall method.

The most important thing to note in the pruning stage is that at any level coefficients are read in order and relevant sections of time series under consideration are passed on the values (as described in section 5). Since the parts of the time series at one level are simultaneously under consideration, a level of coefficients is read as in Step 7 to modify the value of R for that particular level. However, only the coefficient relevant for a particular part is used to adjust the average value avg_S in Step 9.

There is a distance calculation with the reading of coefficients at each level. This may cause the time required to prune the data to increase. So as a tradeoff between time required to prune the data and the amount of data read to do the pruning, a few levels of coefficients can be read initially and bounds can be generated for the segments of the time series. For example, reading the first 4 levels of coefficients (or 8 coefficients) can result in calculation of bounds for 8 consecutive segments of the time series. In the next section we prove how this segmentation is not going to result in the loss of matching candidates.

Once the time series has undergone pruning, there will be situations where there will be adjacent parts of time series which have not been pruned. For example, if a time series S was divided into 4 equal length parts S_1, S_2, S_3, S_4 and only S_3 was eliminated, there is a chance of finding a similar subsequence which lies partly in S_1 and partly in S_2. In such a case, S_1 and S_2 need to be merged. So adjacent parts of time series are merged. Note that those parts that have been eliminated do not have the possibility of having even one point as a part of the subsequence similar to query subsequence. (For proof refer to section) 6.2. This is what we mean by Step 3 in the main algorithm 2.

Algorithm 1. eHaar Pruning Algorithm: $Prune(S, N, R, avg_S, \epsilon)$

INPUT: S, N, R, avg_S, ϵ. Global variables UB_Q, LB_Q
OUTPUT: Global variable $Candidates_S$

$UB_S = avg_S + R, LB_S = avg_S - R$
if $UB_S < LB_Q - \epsilon \parallel LB_S > UB_Q + \epsilon$ **then**
  S does not have a similar subsequence
  break;
else
  if $length(S) > 2 * length(Q)$ **then**
    Read all the coefficients at the next level of S, and assign the specific coefficient for S to C_j
    $R = \sqrt{(R^2 - \sum_k 2^{log_2 N - k} * C_{1+k}^2)}$.
    $avg_{S1} = avg_S - C_j, avg_{S2} = avg_S + C_j$
    $S_1 = S(1 : length(S)/2)$ and $S_2 = S(length(S)/2 + 1 : length(S))$.
    $Prune(S_1, length(S)/2, R, avg_{S1}, \epsilon)$
    $Prune(S_2, length(S)/2, R, avg_{S2}, \epsilon)$
  else
    add S to $Candidate_S$
  end if
end if

Algorithm 2. eHaar Main Algorithm

INPUT: Set of S, Q, ϵ **OUTPUT:** Set of subsequences closest to query Q, mH, Global variable

for each S **do**
  $Prune(S, N, E_S, C_1, \epsilon)$
  Merge adjacent candidates in $Candidate_S$
  for each candidate S_p in $Candidate_S$ **do**
    $DTW_distance(S_p, Q)$
    add nearest subsequence S_{pn} in S_p to maxheap mH
  end for
end for
Find most similar subsequence for Q from maxheap mH

6.2 Proof for Correctness of Pruning

Note that pruning takes place if $UB_S < (LB_Q - \epsilon)$ or $LB_S > (UB_Q + \epsilon)$ conditions are satisfied. The thing we have to prove is that no part of successful candidates lies in the segments eliminated.

Let's assume that one of the points S_{ip} of a successful candidate subsequence lies in the segment (S) under consideration. Let's say that it matches one of the points Q_{jp} of the query subsequence Q. Now,

$$\begin{aligned} distance(S,Q) &= DTW(S,Q) \\ &= \sum_{i,j}(S_i - Q_j)^2 \\ &> (S_{ip} - Q_{jp})^2 \\ &> \epsilon^2 \end{aligned} \tag{7}$$

From the above equation we can see that if even point should belong to the segment S, the distance between S and Q would become larger than ϵ. Hence, proving by contradiction that no point in S can belong to the candidate subsequence and so the segment can be safely eliminated.

6.3 DTW Based Matching

Once the list of candidates is generated, then we need to perform the exhaustive DTW based search of candidates. One of the checks that we perform is that if the candidate has another candidate at next to it at the same level from the same time series. If this is true, then we merge the two candidates. This merging goes on till we reach the end of the time series, or when we do not have any more candidates at the same level from the same time series.

In this step we take advantage of the SPRING method proposed by Sakurai et.al. to get the best matching subsequence. Please note that the method proposed above is the best method to search for matching subsequence as it is linear to the product of query length and the candidate length. Also note that because of having parts of time series being searched for matching subsequences, this algorithm tends to be sub-quadratic in terms of query length and length of the time series.

6.4 Sorting of Matching Subsequences

There is a possibility that every time series has some subsequences in it which are close to the given query subsequence. Using the SPRING method we know the best possible matches in each time series. To maintain a list of the best matches when the search for the subsequences is going on, we use the data structure maxheap. In this data structure we insert the best matching subsequences discovered so far. As soon a subsequence is discovered which is closer to the query subsequence than any subsequence in the maxheap, we delete the corresponding element from the maxheap and insert the latest matching subsequence discovered. The size of the maxheap can be predetermined as well as the range or tolerance level for which candidates are searched.

7 Experimental Results

In this section we experimentally evaluate the performance of eHaar compared to previous subsequence search techniques. We use the SPRING method proposed by Sakurai et.al. in [15] and the LB_Keogh based method proposed in [12], [5] and [17] for comparisons. We do not make comparisons to [14] as it fails to identify matching subsequences which are of lengths different from the length of the query. Also we avoid

comparisons with [16] as it is an approximate method with no guarantees of returning right results for all queries.

Algorithms were implemented in MS Visual C# 2008. Experiments ran on an Intel i5 2.4Ghz machine with 4GB of main memory and 100GB of hard disk space running Windows 7.

7.1 Datasets

We perform experiments on a wide variety of datasets characterized by diverse features. Stock contains 500 time series of length 256; it consists of opening prices of 500 stocks in NYSE during a financial year. Our next data sets originate from the UCR archive[1], and are created by sampling a collection of constant-length subsequences from very large time series. Arcene derives from a set of training and test data with 800 sequences of length 4096.

The largest dataset is the RandomWalk dataset created using the Random Walk time series generator. The seed of the generator was set at 1416. We produced data sets of 256-length time series, ranging from $10,000$ to $1M$. Please note that the size of the RandomWalk dataset is approximately 5 GB which is almost 100 times greater than the largest possible dataset reported in the literature ([16])so far. The query subsequences of different lengths, ranging from length 8 to length 64, were extracted randomly from the respective databases. We chose to ran 10 queries per database with queries varying in length. Results reported are averaged over these queries.

7.2 Performance Measures

The efficiency of the methods can be calculated using the following measures:

1. **DTW cell cost:** For each query Q, the DTW cell cost is the number of cells [i][j] visited by the method to derive the result. This basically reflects the cost of running the algorithm. For the entire set of queries, the average value of DTW cell cost is reported. For the SPRING method this number is the product of query length and database length.
2. **Total terms accessed:** This basically measures the amount of information from the time series datasets that needs to be accessed by the algorithm. It is basically an I/O cost which is used to show the tremendous amount of information that needs to be read for subsequence search. When the dataset size is large, data cannot be stored in the main memory and so it needs to be accessed from storage devices. This deteriorates the performance of the algorithm. Since the size of datasets is increasing, this measure is quite important for comparisons of methods.
3. **Retrieval runtime cost:** For each query Q, given a method, the retrieval runtime cost is the total retrieval time for the matching subsequences using that method. For eHaar we include the time taken to prune the unwanted parts of time series data. Although runtime can depend on a variety of factors like cache size etc., we try to provide a level playing field for all the three methods.

[1] See http://www.cs.ucr.edu./~eamonn/time_series_data

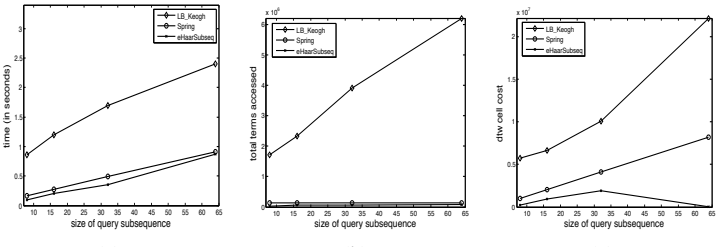
7.3 Results

We compare the performance of eHaar with LB_Keogh based method and SPRING. The first set of experiments were conducted on stock dataset with the three measures being reported. The experiments were run for sets of different length of query subsequences. 10 queries each of length 8,16,32 and 64 were run on the dataset and the results have been included in the figure 3. We observe a clear advantage for eHaar over other methods as it does not read the full data and prunes data effectively.

The advantage of SPRING over all methods for subsequence search is that it always accesses fixed number of terms and its DTW cell cost is constant. This is an asset in situations were subsequences are very close to the entire subsequence and hence searching the whole time series is essential. We perform our experiments with different sets of queries where in one set the size of the queries is constant. The sets of queries were of length 8, 16, 32 and 64. In a small dataset like Stock both SPRING and eHaar appear competitive in terms of the 3 parameters (refer to Figure 3).

Moving on to a slightly larger dataset arcene, there is a competition between the methods in the area of total terms accessed. However because the amount of DTW calculations are less for eHaar, it scores over SPRING and the improvement is clearly shown in terms of speedup (Figure 4 (a)). This is so because the terms are accessed in eHaar only for pruning and not for distance calculations in the case of SPRING.

In a Z-normalized dataset RandomWalk, where the time series data has been normalized and also the queries, a marked improvement of over 30% is observed in the case of eHaar. The advantage of accessing less terms leads to an improvement over DTW cell cost and so the final result is that eHaar needs less time to search the subsequences. In case of RandomWalk we had a set of queries with their lengths varying from 8 to 128. Repeated experiments were performed on different sizes of RandomWalk dataset, ranging from a size of 10,000 to 1 million time series.

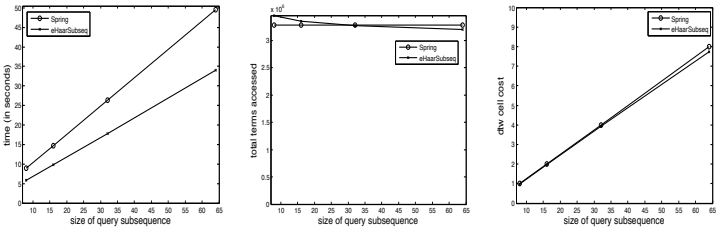


(a) (b) (c)

Fig. 3. Stock Dataset

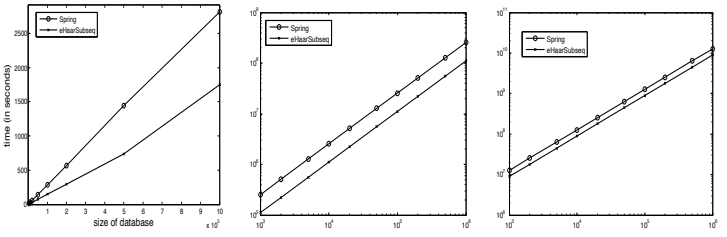
For experiments on larger datasets we forego experiments on LB_Keogh as the time taken and the terms used is too large for comparisons with SPRING and eHaar. Arcene dataset has time series with large lengths and the queries effectively show the superiority of eHaar over SPRING in long time series datasets (Figure 4).

We further carried out experiments on a very large dataset Random Walk which has 1 million time series data. This is a disk based data and cannot be stored in memory.



(a) (b) (c)

Fig. 4. Arcene Dataset



(a) (b) (c)

Fig. 5. Random Walk Dataset

We perform experiments on all the three performance measures. Figure 5 shows the results. The results show the advantage over SPRING because it requires less processing, accesses less data and hence needs less time for search.

8 Related Work

One of the first works on subsequence matching has been proposed by Faloutsos et.al. in [4]. This work basically opts for an exhaustive search of subsequences, generating first the exhaustive list of candidates, indexing them and then carrying out the search for query subsequence. It extracts subsequences of a fixed length from a time series sequence using sliding windows. The subsequences extracted are transformed into a low-dimensional point and are indexed as a point in a R^*-tree. For a given query sequence, subsequences of same length are extracted, however with the subsequences being disjoint as opposed to sliding window extraction. Subsequently the subsequences extracted are transformed into low-dimensional points and the search for matching points is carried out in the R^*-tree, given the range and tolerance. A list of candidates is extracted from the R^*-tree and the false positives are eliminated to obtain the final set of matching subsequences.

Some works try to improve the performance of subsequence matching by using different window construction methods from [4]. DualMatch [8] divides a time series sequence into a set of disjoint windows. Instead of extracting disjoint subsequences from the query sequence, sliding windows are used to extract subsequences. Another approach proposed by the same authors tries to generalize the concept of sliding windows and disjoint windows. GeneralMatch [9] proposes the use of J-sliding windows and

J-disjoint windows and a subsequence matching method based on these windows. Both the methods, however, use the same R^*-tree based indexing and search of subsequences.

[10] proposes to reduce the DTW based computations by reducing the length of the time series sequences and the query sequence. From the points of the time series monotonically increasing or decreasing segments are formed and then DTW distances are calculated using these reduced representations. However, the method is applicable to only one-dimensional time series. [11] proposes to use constant length segments to approximate a time series sequence, instead of monotonic sequences. Basically a fixed number of points in the time series are replaced by their average value.

An LB_Keogh [12] based subsequence search method has been proposed by Wong et.al. in [5] The approach is based on the sliding window and the sliding window constraints. The lower bound function is used in the sliding windows. The problem with this method is that the bounds are loose and only the prefixes of the possible subsequences are indexed. [13] proposes an index structure based on suffix trees for subsequence matching. The problem with this method is that the complexity of the search process is still quadratic.

Han et.al. propose an exact ranked based subsequence matching method in [14]. It also uses the lower bound LB_Keogh proposed in [12] after breaking the query sequence and the time series sequence into segments. However, this method suffers from the following drawbacks. First, it only considers those subsequences which are equal in length with the query sequence.Also the method is only applicable to constrained DTW. This means that the warping path has to stay close to the diagonal.

One of the breakthrough works in the area of subsequence matching has been the SPRING method proposed by Sakurai et.al. in [15]. Although the complexity of this DTW based algorithm is still quadratic, it does away with the generation of all possible subsequences typically carried out before the matching process. This is possible as it incrementally calculates the DTW distance. Although the method is mainly proposed mainly for streaming data, it is useful for search in time series databases.

[16] describes EBSM (Embedding-Based Subsequence Matching) method for approximate subsequence matching. It converts subsequence matching to vector matching using an embedding. This embedding maps each database time series into a sequence of vectors. The embedding is computed by applying full dynamic time warping between reference objects and each database time series. At runtime, given a query object, an embedding of that object is computed in the same manner, by running dynamic time warping between the reference objects and the query. Comparing the embedding of the query with the database vectors is used to efficiently identify relatively few areas of interest in the database sequences. Those areas of interest are then fully explored using the exact DTW-based subsequence matching algorithm. The drawback of this method is that it does not guarantee correct results for queries.

8.1 Discussion

Most of the above works require the generation of subsequences from all the time series. In some cases the number of subsequences is $N(L - w + 1)$ while in some cases it is NL/w where w is the length of the subsequence, N is the number of time series sequences and L is the length of the time series sequence. In this work, we propose that

if proper bounds around sequences can be developed, a lot of time series, which do not have candidates matching with the query, can be directly eliminated without analyzing the subsequences in it. This premise based on eHaar is the key to exact pruning of the candidates without analyzing them, even without analyzing the details of the parts of the time series just on the basis of energy information.

9 Conclusion

This paper proposes a change of approach to subsequence search in time series data. eHaar uses minimum mount to information to prune parts or whole of time series sequences which do not have the possibility of having subsequences within a given range of the query subsequence. It only avoids generating an exhaustive list of possible candidates like SPIRNG, it also goes one step ahead by pruning time series sequences without reading them completely using the upper and lower bounds developed for a time series and its parts. This effectively speeds up the process manifolds and at the same time maintains the accuracy of the results.

The experiments prove the efficiency of the method with respect to the state of the art. In the future work we aim to improve the efficiency of the method by tightening the bounds and also develop effective methods of DTW distance calculations for eHaar based on wavelet coefficients.

References

1. Tan, P., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Education Inc., London (2006)
2. Han, J., Kamber, M.: Data Mining, Concepts and Techniques. Morgan Kaufmann Pub., San Francisco (2006)
3. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730. Springer, Heidelberg (1993)
4. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: SIGMOD (1994)
5. Wong, T., Wong, M.: Efficient subsequence matching for sequences databases under time warping. In: IDEAS (2003)
6. Lim, S.-H., Park, H., Kim, S.-W.: Using multiple indexes for efficient subsequence matching in time series databases. In: Li Lee, M., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, pp. 65–79. Springer, Heidelberg (2006)
7. Loh, W.K., Kim, S.-W., Whang, K.-Y.: A subsequence matching algorithm that supports normalization transform in time-series databases. In: DMKD (2004)
8. Moon, Y.S., Whang, K.Y., Loh, W.K.: Duality-based subsequence matching in time-series databases. In: ICDE (2001)
9. Moon, Y.S., Whang, K.Y., Han, W.S.: General match: A subsequence matching in time-series databases. In: SIGMOD (2002)
10. Park, S., Kim, S., Chu, W.W.: Segment-based approach for subsequence searches in sequence databases. In: SAC (2001)
11. Keogh, E., Pazzani, M.: Scaling up dynamic time warping for data mining applications. In: KDD (2000)
12. Keogh, E.: Exact indexing of dynamic time warping. In: VLDB (2002)

13. Park, S., Chu, W.W., Yoon, J., Won, J.: Similarity search of time-warped subsequences via a suffix tree. Information Systems (2003)
14. Han, W.S., Lee, J., Moon, Y.S., Jiang, H.: Ranked subsequence matching in time series databases. In: VLDB (2007)
15. Sakurai, Y., Faloutsos, C., Yamamuro, M.: Stream monitoring under the time warping distance. In: ICDE (2007)
16. Athitsos, V., Papapetrou, P., Potamias, M., Kollios, G.: Approximate embedding-based subsequence matching of time series. In: SIGMOD (2008)
17. Niennattrakul, V., Ratanamahatana, C.A.: Meaningful subsequence matching under time warping distance for data stream. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 1013–1020. Springer, Heidelberg (2009)
18. Lin, S.-C., Yeh, M.-Y., Chen, M.-S.: Subsequence matching of stream synopses under the time warping distance. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010. LNCS, vol. 6119, pp. 354–361. Springer, Heidelberg (2010)
19. Chan, K.-p., Fu, A.W.-c., Yu, C.: Haar Wavelets for Efficient Similarity Search of Time-series: With and Without Time Warping. IEEE TKDE 15(3), 685–705 (2003)
20. Stollnitz, E.J., DeRose, T.D., Salesin, D.H.: Wavelet for Computer Graphics: A Primer Part 1. IEEE CGA 15(3), 76–84 (1995)

Optimizing Predictive Queries on Moving Objects 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. Advanced wireless communication and positioning technology has enabled a new series of applications, such as the intelligent traffic management system. It can be envisioned that the traffic management systems will have a great impact on our daily life in the near future. This paper aims to tackle one class of queries to be supported by such systems, predictive line queries. The predictive line query estimates amount of vehicles entering a querying road segment at a specified future timestamp and helps query issuers adjust their travel plans in a timely manner. Only a handful of existing work can efficiently and effectively handle such queries since most methods are designed for objects moving freely in the Euclidean space instead of under road-network constraints. Taking the road network topology and object moving patterns into account, we propose a hybrid index structure, the R^D-tree, which employs an R*-tree for network indexing and direction-based hash tables for managing vehicles. We also develop a ring-query-based algorithm to answer the predictive line query. We have conducted an extensive experimental study which demonstrates that our approach significantly outperforms existing works in terms of both accuracy and time efficiency.

1 Introduction

Advances in wireless devices and positioning systems have enabled the tracking of moving objects such as vehicles equipped with GPS, and fostered a series of new applications. An important application which may have a great impact on our daily life is the intelligent traffic management system. The intelligent traffic management system is expected to improve travel efficiency by means of traffic monitoring and prediction, route redirection in case of congestion, etc. With the aid of such system, users should be able to conduct a query like "How will the traffic condition be on Highway 44 near St. Louis in half an hour?". An example query is shown in Figure 1, where a user is interested in the traffic condition of the highlighted road segment in the near future. The query result will help user to make adjustment on his/her travel plan. We term such queries as *predictive line queries*. In this work, we aim to develop efficient solutions for this type of queries.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 247–261, 2011.

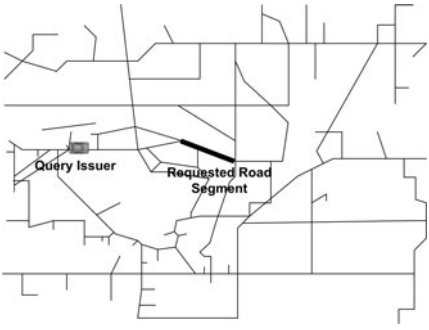
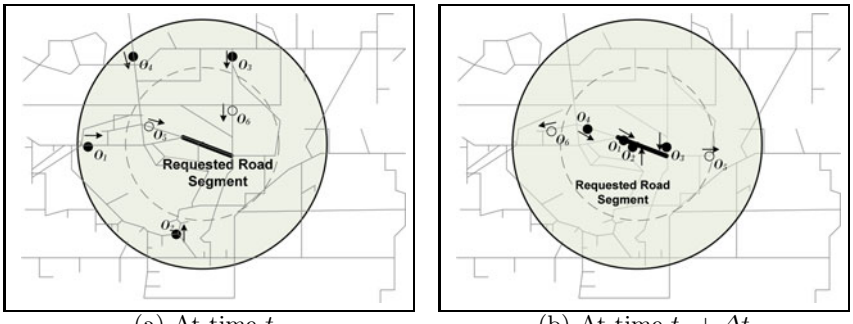


Fig. 1. An Example of Predictive Line Query

Despite extensive research in moving object databases, handling predictive line queries is still a substantial challenge. This is because most moving object management techniques [12, 20, 23, 24] model objects moving freely in Euclidean space but do not consider road-network constraints. Euclidean space based approaches fall into two categories: i) Those that rely on solutions that store a snapshot of object's position at each timestamp [12, 20]. This approach is not able to support any predictive queries based on traffic prediction. ii) Those that rely on solutions that represent object's position using a linear function [4, 9, 15, 18, 23, 24, 28]. To be more specific about the later category, object's future positions are predicted by assuming that the object moves along a straight line at the latest updated velocity. This is however not realistic under the road-network constraints; roads are more often curvy than being straight lines. Therefore in general, queries generated based aforementioned approaches lack accuracy in terms of predictive line queries.

However, some of the recently proposed indexing structures, handling moving objects on road networks rather than Euclidean space, only support queries on historical or current positions of objects, but cannot provide traffic forecast [2, 7, 28]. Current indexes that may support the predictive line query, such as the R-TPR$^{\pm}$ tree [6], issues a range query defined using a circle or a rectangle covering the querying road segment. The size of the range is determined by the maximum possible traveling speed in order to cover all objects that may enter the query road segment at the query time. However, this approach is very inefficient since it visits many unnecessary objects such as objects in the query range moving away from the query road segment. Figure 2 depicts a scenario explaining this situation. Figure 2(a) shows a snapshot of objects' positions at timestamp t_0 when a query is issued, and Figure 2(b) shows positions at the query time $t_0+\Delta t$. The arrow besides the object indicates its moving direction. As can be seen from the figure, objects O_5 and O_6 which are closer to the query road segment e_q (highlighted by the bold line) at t_0, already pass e_q at query time $t_0+\Delta t$. Only objects which are not too far away or too close to the query road segment, as those (denoted by black points) located in the ring area, may be on the query road segment at the query time.



(a) At time t_0 (b) At time $t_0 + \Delta t$

Fig. 2. Unnecessary Moving Objects Considered in the Range Query

In this paper, our goal is to develop efficient and effective indexing and querying techniques to support predictive queries like the predictive line queries, on moving objects under the road-network constraints. In particular, we propose a new index structure, called R^D-tree, where D stands for direction. The R^D-tree employs the R*-tree to index road segments, and arrange objects on each road segment based on their traveling directions. To answer the predictive line queries, we estimate object's future traveling routes following the road networks, and leverage the concept of *ring* queries to constraint our search within the set of objects which have high probabilities to enter the query road segment at the query time. In this way, we largely reduce the amount of false positives and hence significantly improve the performance. We conducted an extensive experimental study and the results demonstrate that our approach outperforms the R-TPR$^{\pm}$ tree in terms of both efficiency and accuracy.

The rest of this paper is organized as follows. Section 2 reviews related works on moving object management. Section 3 presents our proposed index structure and query algorithms. Section 4 reports the experimental results. Finally, Section 5 concludes the paper.

2 Related Work

Moving object management techniques can be classified into two main categories: real-time moving object database systems, and historical moving object database systems [16,22]. In what follows, we review mainly works in the first category since our work falls into this category.

The main challenge in real-time moving object databases is due to the frequent updates of object positions. Thus, the emphasis of moving object indexing is on efficient handling of large amounts of updates in a timely fashion in order to provide up-to-date query results. Works in this category can be further divided into two subcategories: (i) handling objects moving in Euclidean space; and (ii) handling objects moving on road networks. As mentioned earlier in the introduction, indexes [4,9,15,23,24] that model objects moving freely in Euclidean

space are not capable to provide accurate query results under road-network constraints. For this reason, this discussion is focused on the second subcategory - handling objects moving on road networks.

The following work considered objects on road networks. Shahabi et al. proposed a road network transformation method [19] to make it easier to apply the traditional nearest neighbor query algorithm adopted in Euclidean space. Kim et al. [11] proposed an indexing structure called IMORS which stores road information using the R*-tree. The leaf nodes of the R*-Tree contains objects on the road segments. In [26], Wang et al. proposed a dual-index which employs a disk-based structure, the R-tree, to store static road network, and an in-memory grid structure to store object positions. In-memory data structure is fast for position updates yet lack the scalability to deal with a large number of objects. Aiming at improving update performance, Bok et al. [2] proposed an IONR-tree which captures the connectivity of road networks. The basic idea is to store multiple edges connected by the same intersection node in the same index node so that some object updates can be done in the same index node when the objects travel from current edge to the neighboring edge. The literature also discusses of the work with focus on query processing rather than indexing structures. Mouratidis et al. [14] proposed a method to continuously monitoring k nearest neighbors in road networks. They assume that both road network and objects are stored in memory. As a variant of k nearest neighbor (k-NN) queries, Qin et al. [17] proposed the continuous aggregate k-NN queries. Sun et al. [21] and Li et al. [8] deal with continuous range queries and reverse nearest neighbor queries in road networks, respectively. Lai et. al [13] studied the continuous density queries in road networks, where density computation is determined by the length of the road segment and the number of objects on it. In addition, some distributed indexing scheme [10,27] have been proposed, which rely on peer-to-peer communication to gather real-time traffic information. In contrast to the aforemention efforts, which support only current or/and continuous queries, our work also supports predictive queries.

Our proposed index may look similar to IMORS. However, unlike IMORS, which groups objects based on their current locations, our proposed index stores moving objects based on their traveling destinations. Also, we support predictive queries but IMORS only deal with queries on current positions. The closest related work to our approach is the R-TPR±-tree [6], which supports predictive queries in road networks. Thus, we describe it in more details as follows. The R-TPR±-tree consists of an R*-tree and multiple TPR-trees [24]. The R*-tree indexes road-networks. Each road segment, stored in the R*-tree, maintains a modified TPR-tree, namely TPR±-tree, to index moving objects. Objects are divided into two groups based on their moving directions along the road segment. The root of the TPR±-tree has two children TPR-trees, one for each direction. This method reduces the expansion of the minimum bounding rectangles and hence reduces the update cost compared to the original TPR-tree proposed in [24]. An algorithm has been developed to estimate objects that may enter the query road segment at a future timestamp. The algorithm performs a range

query to retrieve potential road segments that may contain objects in the query results. Then for each road segment, it checks if the earliest entered object will be traveling on the same road segment at the query time. If so, all objects on this road segment are considered not to be able to enter the query road segment. Similarly, if the lastly entered vehicle has passed its original road segment at the query time, all objects on that road segment will be added to the query results. Such estimation is not very accurate since it does not consider individual object's future position. In addition, we believe that the experimental results, as reported in [6], are based on unpractical assumptions. For example, predicting object positions up to 30 seconds from current timestamp is not a practical assumption, since objects either stay on the same road segments or neighboring segment. Such short predictive time window may not be useful in real world applications since users will not have sufficient time to plan for a new route after they know the traffic conditions.

The literature has addressed a few works for predictive queries [5,25]. However, these solutions are based on in-memory data structures, which may not scale up well with datasets.

3 The R^D-Tree

In this section, we first present the data structure of our proposed R^D-tree and then describe the algorithms for the predictive line query and index maintenance.

3.1 The Index Structure

The R^D-tree indexes two types of data: road-network information and object location 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 and $e = \{v_1, v_2\}$, 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. A moving object, vehicle, O is represented by the tuple $\{vId, x_1, y_1, e_c, e_d, speed, gd, t\}$, 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_d is the next road segment that the vehicle is heading to, and gd is the vehicle's traveling destination. Here, we assume that most moving objects are willing to disclose their tentative traveling destinations to the server in order to obtain high-quality services, however the destination may change during the trip.

The R^D-tree is composed of an R*-tree [1] and a set of hash tables. Leaf nodes in R*-tree pointing to hash tables representing vehicles of each road segment. Figure 3 illustrates the overall structure of the R^D-tree. The road-network information is indexed by the R*-tree. Each entry in the non-leaf node is in the form of $(node\_MBR, child\_ptr)$, where $node\_MBR$ is the minimum bounding

[1] Road segments and edges may be used interchangeably throughout this paper.

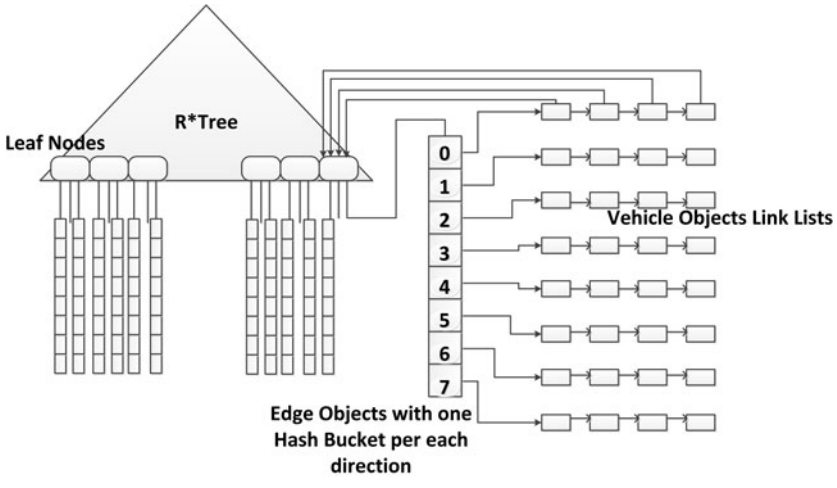
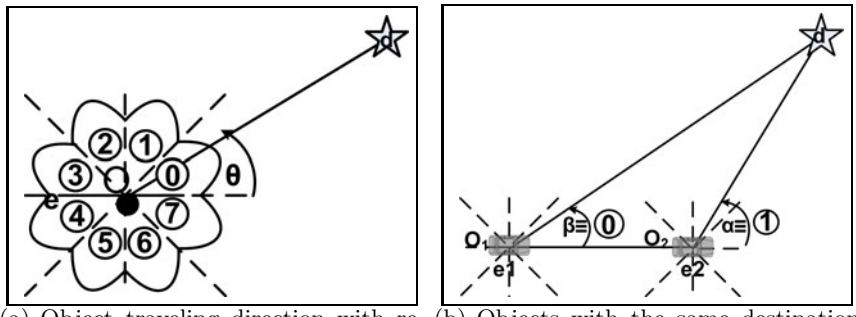


Fig. 3. Index structure for the road network

rectangle (MBR) covering the MBRs of all entries in its children pointed by $child\_ptr$. Each entry in the leaf node is in the form of $(edge\_MBR, obj\_ptr)$, where $edge\_MBR$ is the MBR of a road segment and obj_ptr links to a hash table storing objects moving on this edge.

Each hash table has N_d slots, where N_d is the number of traveling directions. In the example showing in Figure 3, N_d is equal to 8. Moving objects with similar traveling directions are hashed to the same slot and stored as a link list. For easy update, each object also has a pointer directly linked to the edge that it is currently located.

The critical issue to construct the hash table is to determine an effective hash function which groups objects with similar traveling directions. The object traveling direction is determined by the angel between the horizontal line and the line connecting the object current position to its destination. For example, in Figure 4(a), object O's traveling direction is indicated by θ, and its destination is indicated by the star. By equally partitioning the 360 degree into 8 directions, object O's traveling direction falls into the direction 0 which can be treated as a hash value. This strategy will result in following issue: As shown in Figure 4(b), two objects O_1 and O_2 moving on the same road segment with the exactly same destination obtain two different directions 0 and 1 respectively, simply because the minor difference between their current positions. From the querying perspective, these two objects are expected to be stored together since they are very likely to have similar or the same travel path. Therefore, we make the following adjustment to ensure that such objects will obtain same hash values. Instead of using objects' current positions, we use the middle point of the road segment to compute the angle. The formal definition of an object's hash value is given below.



(a) Object traveling direction with respect to current road segment

(b) Objects with the same destination but different traveling directions

Fig. 4. Object Traveling Direction

Definition 1. *Let O be a moving object which is currently on road segment e with traveling destination gd. Let θ denotes the angle between the horizontal line of the coordinate system and the line connecting gd and the midpoint of e. O's hash value is defined by Equation 1, where N_d is the number of buckets in a hash table.*

$$H(O) = \theta \mod \frac{360}{N_d} \tag{1}$$

As an example, when θ is 30 degrees, and N_d equals to 8, $H(O)$ will be $30 \mod (360/8) = 0$. That means object O will be stored in the first slot of the hash table.

3.2 Algorithms for Predictive Line Queries

Our R^D-tree can support traditional types of queries, such as range queries and k nearest neighbor queries. Concerning the road network constraint, we refine the range query to the line query. Instead of locating objects in a certain rectangular or circle range, the line query estimates the moving objects which may enter the query road segment (i.e., a line) at the query time. The motivation of such line query is that people are usually more interested in the traffic condition of a particular road that they need to pass by, rather than the traffic condition of a wide range which may contain roads irrelevant to the query issuers' traveling routes. The formal definition of the predictive line query is as follows.

Definition 2. *[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.*

Our algorithm for the predictive line query consists of two phases. The first phase is a filtering phase which retrieves candidate objects using a *ring query* (defined in the later text). The second phase refines the results by estimating the candidate objects' traveling routes.

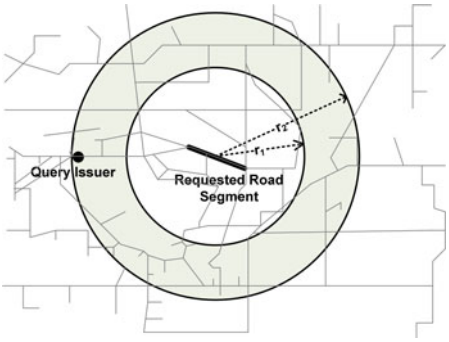


Fig. 5. The initial filtering with a *Ring query*

Given a predictive line query, we first compute its corresponding ring query. The ring query aims to define a more constraint search range than the general rectangular or circular range queries so that fewer intermediate results are generated. The basic idea is to find the current positions of the furthest vehicle and closest vehicle which may enter the query road segment at the query time, and then use their current distance to the query road segment to define concentric circles as the query ring. More specifically, the furthest candidate vehicle is currently moving at distance $v_{max} \cdot (t_q - t_c)$ from e_q, while the closest candidate vehicle is currently moving at $v_{min} \cdot (t_q - t_c)$, where v_{max} and v_{min} are the maximum and minimum speed limits respectively. The area covered by the ring query is $\pi(v_{max}^2 - v_{min}^2) \cdot (t_q - t_c)^2$, while that of the range query is $\pi v_{max}^2 \cdot (t_q - t_c)^2$. The smaller range given by the ring query reduces the number of moving objects to be accessed in the index. Graphical explanation of the ring query is illustrated in Figure 5. Its formal definition is given in Definition 3.

Definition 3. *[Ring Query] A ring query* $RQ = (e_q, r_1, r_2)$ *retrieves moving objects whose current locations are in the ring defined by the concentric circles with the mid point of the query road segment* e_q *as center and* r_1 *and* r_2 *as radius, where* $r_1 = v_{min} \cdot (t_q - t_c)$ *and* $r_2 = v_{max} \cdot (t_q - t_c)$.

Once the query ring is determined, we start the search in the R^D-tree to find road segments that intersect with the query ring. For each road segment in the query ring, we further check its hash table to find objects currently moving on it. In fact, we do not need to access the entire hash table. We only access the hash buckets which contain objects with traveling directions toward the query road segment. In particular, we first compute the angle θ_q between the horizontal line and the line connecting the mid points of the current road segment and the query road segment. Then we plug in θ_q to Equation 1 to obtain a hash value H_q. Figure 6 illustrates the idea, where the hash value is 0. From Figure 6, we also observe that the query θ_q is located at the border of the hash bucket 0. Thus, to obtain more accurate query results, we consider one more bucket adjacent to H_q when θ_q is close to the border with less than θ_x degree (we set it to 15 degree as default). In the example, both buckets 0 and 1 are considered in the query.

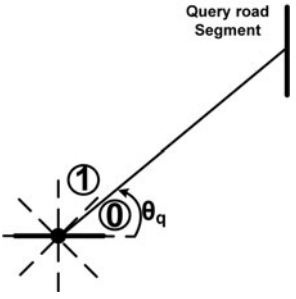


Fig. 6. An example for marginal query angle selection

After obtaining a set of candidate objects from the ring query, the second phase of the query processing eliminates objects which are not possible to enter the query road segment by examining object tentative traveling routes. When an object is initially registered in the system or issues an update of its destination, we compute the shortest route to its destination. During the query, we check if the shortest route of the candidate object contains the query road segment at the query time. If so, the candidate object will be included in the final result. It is worth noting that being a prediction, the query results may not be 100% accurate. The query algorithm is summarized in Algorithm 1.

In Algorithm 1, when a user (moving object) sends a query request, he/she does not need to always specify the query time. We estimate the time taken for the query issuer to enter the query road segment as the query time t_q when

Algorithm 1. Algorithm for Predictive Line Query
Inputs: loc_q – current location of the query issuer, e_q – query road segment, t_q – query time, t_c – Query issuing time
Output: $Result$ – a set of objects that may be on e_q at t_q

1: **if** t_q =NULL **then**
2: $t_q = timeToEnter(v, e, t_c)$
3: **end if**
4: $Result = \emptyset$
5: $Edges = RingQuery(e_q, t_q, v_{min} \cdot (t_q - t_c), v_{max} \cdot (t_q - t_c))$
6: **if** $(Edges <> null)$ **then**
7: **for** each $e_i \in Edges$ **do**
8: $Direction = getDirection(e_i, e_q)$
9: $Result = Result \bigcup getVehicles(e_i, Direction)$
10: **end for**
11: **end if**
12: **for** each object o_i in $Result$ **do**
13: **if** *not* $getVehiclesContainPaths(e_q, o_i, t_q)$ **then**
14: $Result = Result - \{o_i\}$
15: **end if**
16: **end for**

it is not provided. Line 5-11 are the first phase. The function 'getDirection()' returns two consecutive hash values with the hash value of the direction to the query road segment in the middle. The function 'getVehicles()' checks the hash table of the particular edge and only retrieves moving objects with the hash values given by 'getDirection()'. Candidate objects are stored in a set *Result*. Line 12-16 are the second phase. The estimated traveling route of each candidate object in *Result* is checked. If the traveling route does not contain the query road segment at the query time, the object will be removed from *Result*.

3.3 Insertion and Deletion in the R^D-Tree

An object position update can be seen as a deletion followed by an insertion. An update request contains the object ID, previous road segment and destination, current road segment, current position and velocities, and new destination if there is any change. First. we search the R^D-tree to find the leaf node containing the previous road segment that the object was on. Once the leaf node is located, we compute the hash value according to Definition 1 using the object's previous road segment and destination. Then, we locate the corresponding hash bucket to find the object. If the object's previous and current road segment are the same as well as the traveling destinations, we just need to update the object position and velocity information in the hash bucket. Otherwise, we delete the object's old information and perform the following insertion steps. We check if the object is still on the same road segment but with a new destination. In this case, the update is conducted under the same leaf node. If not, we need to search the R^D-tree to locate the leaf node containing the current road segment. After that, a hash value is computed based on the new road segment and destination, and the object current information is inserted to the corresponding hash bucket linked to the leaf node.

4 Performance Study

We used the Brinkhoff's generator [3] to generate moving objects on real road maps of states in the U.S. The number of moving objects in each dataset ranges from 10K to 100K. The object speeds range from 30mph to 60mph. The California state map was used as default, which contains 17,702 road segments. We generated predictive queries by randomly selecting query road segment and predictive time length.

We compared the performance of our R^D-tree with the R-TPR±-tree [6] which supports predictive queries on moving objects under road network constraints. In the experiments, we evaluated three factors: the number of moving objects, the predictive time length, and the road topology. The performance was measured in terms of I/O cost (the number of disk-page accesses) and accuracy. The accuracy was examined 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. Parameters and their values are summarized in the Table 1, where default values are highlighted in bold.

Table 1. Parameters and Their Values

Parameters	Values
number of moving objects	10K, 20K, ..., **50K**, 60K, ..., 100K
predictive time length (in minutes)	10, 20, **30**, 40, 50, 60
road maps	CO, AR, NM, **CA (California)**

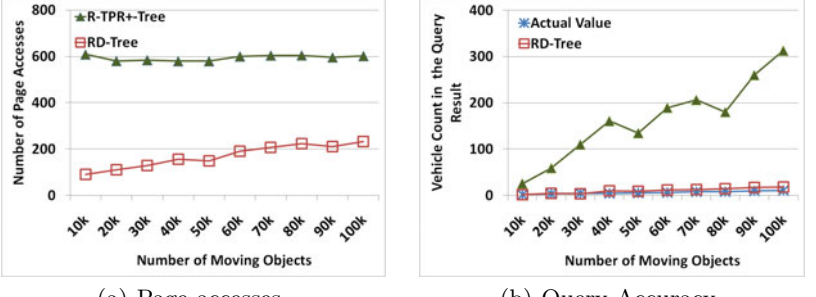
4.1 Effect of the Number of Moving Objects

Both the R-TPR±-tree and the R^D-tree were tested for different sizes of moving object datasets generated using the default road map–the CA map. Figure 7(a) and 7(b) show the performance comparison of the two trees when the predictive time length is 30 minutes. From Figure 7(a), we can see that the proposed R^D-tree requires about half less page accesses than the R-TPR±-tree. The reasons are mainly two-fold. First, the R^D-tree uses the ring query to retrieve candidate objects which are usually less than objects retrieved using the range query. Second, the R^D-tree arrange objects according to their traveling directions. The query only need to check objects that probably will be on the query road segment, i.e. those objects heading the query road segment.

With respect to the accuracy, the R^D-tree also significantly outperforms the R-TPR±-tree as shown in Figure 7(b). In particular, the number of query results returned by the R^D-tree is very close to the actual number of objects on the query road segment. This indicates the effectiveness of our approach. The query results obtained using the R-TPR±-tree contains too many irrelevant objects since the query algorithm of the R-TPR±-tree is mainly suitable for very short predictive time length.

4.2 Effect of the Predictive Time Length

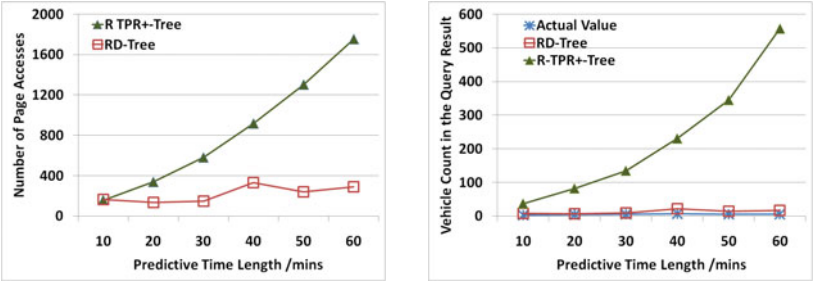
Next, we studied the effect of the predictive time length by varying it from 10 minutes to 60 minutes. As shown in Figure 8(a), both trees access more disk pages when the time length increases. This is because the longer time to look



(a) Page accesses (b) Query Accuracy

Fig. 7. Query Performance when Varying the Number of Moving Objects

into the future, the bigger query range needs to be checked, which results in more page accesses. We also observed that the query cost using the R^D-tree only slightly increases, whereas the query cost using the R-TPR±-tree increases drastically. The advantage of the use of ring query by the R^D-tree is more prominent when the query time length is longer. The area of a query ring increases less significantly than the area of a query circle. Therefore, the number of objects that need to be retrieved in the R^D-tree also increases very slowly.



(a) Page accesses (b) Accuracy

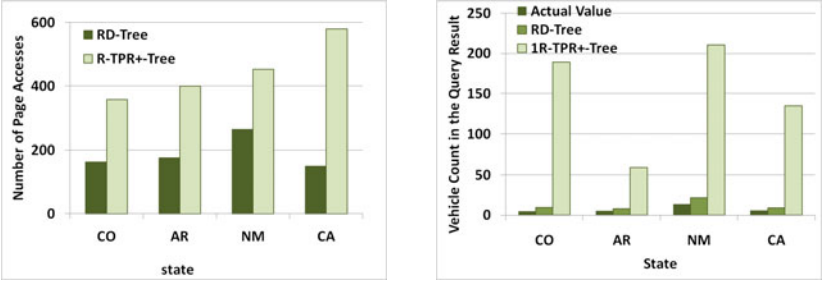
Fig. 8. Effect of the Predictive Time Length

Figure 8(b) illustrates the accuracy. The results obtained by the R^D-tree query algorithm are very close to the actual values, and the accuracy is relatively stable for different query time lengths. The minor inaccuracy may be caused by the difference of the estimated traveling routes and the actual routes taken by some objects. The accuracy in the R-TPR±-tree is much lower compared to the R^D-tree. Especially when the query length is longer, e.g., 60 minutes, the R-TPR±-tree query algorithm returns more than 10 times objects than the actual number of objects on the query road segment. The R-TPR±-tree query algorithm works well when the predictive time length is extremely short so that the query range mainly covers road segments next to the query road segment, and objects in the query range can at most move to the next road segment at the query time. When the predictive time length is long, such estimation introduces lots of errors.

4.3 Effect of the Road Topology

We also evaluated the effect of the road topology by testing different road maps: Colorado (CO), Arkansas (AR), New Mexico (NM), and California (CA). The average road segment length in these maps are different, which are 0.152 miles in CO, 0.101 miles in AR, 0.92 miles in NM, and 0.81 miles in CA. Figure 9 shows the results for the R^D-tree and the R-TPR±-tree. Observe that the R^D-tree significantly outperforms the R-TPR±-tree in all cases in terms of both query

efficiency and query accuracy. Moreover, the performance of the R^D-tree is relatively independent of the road topology, while the R-TPR±-tree performs worse when the road segment becomes shorter. In the R^D-tree, longer road segments result in more objects per hash bucket, and hence slightly affects the performance. In contrast, the R-TPR±-tree performs better for maps with lengthier road segments. The possible reason is that each TPR-tree in the R-TPR±-tree groups objects better when the road segment is longer.



(a) Page access (b) Accuracy

Fig. 9. Effect of Road Topology

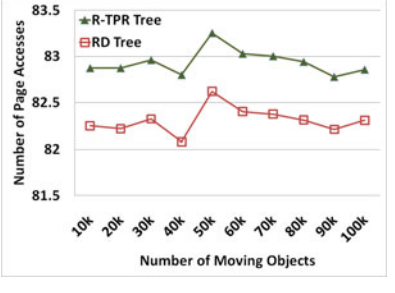


Fig. 10. Update Cost

4.4 Update Cost

Finally, we examined the update cost in the R^D-tree and the R-TPR±-tree. Figure 4.3 shows the average cost after all objects have been updated once. In the experiment, 50 pages of buffer was used. We can see that the two trees perform similarly. This is possibly due to the similarity of the update algorithms. In both trees, the update cost includes two portions. One is for the search in the R*-tree to locate the road segment, and the other is for the search in either the hash table in the R^D-tree or the TPR-tree in the R-TPR±-tree to find the actual object.

5 Conclusion and Future Work

We presented a hybrid indexing structure, the R^D-tree, to support predictive queries on objects moving under the road network constraints. The R^D-tree employs an R*-tree to store road segments, and organizes objects based on their traveling directions. We developed an efficient query algorithm to estimate objects that may enter the query road segment at a future timestamp. Our query algorithm uses a new pruning technique, the ring query, to reduce the amount of intermediate results so as to improve the query performance. Compared to existing approaches, proposed approach achieves significant performance improvements in terms of both query efficiency and accuracy.

The experiments will be repeated and further extended with real datasets and other traffic effecting parameters - weather and accidents - are to be considered.

Acknowledgement. The authors would like to thank Jiamin Lu, one of the authors of [6] for detailed explanations of their algorithm. This work was supported in part by the U.S. National Science Foundation under Grant No. IIS-0324835.

References

1. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The r*-tree: An efficient and robust access method for points and rectangles. In: Proceedings of ACM SIGMOD International Conference on Management of Data (1990)
2. Bok, K.S., Yoon, H.W., Seo, D.M., Kim, M.H., Yoo, J.S.: Indexing of continuously moving objects on road networks. IEICE - Trans. Inf. Syst. (2008)
3. Brinkhoff, T.: A framework for generating network-based moving objects (2004)
4. Chen, S., Ooi, B.C., Tan, K.-L., Nascimento, M.A.: St2b-tree: A self-tunable spatio-temporal b+-tree index for moving objects. In: Proceedings of ACM SIGMOD International Conference on Management of Data (2008)
5. Dittrich, J., Blunschi, L., Salles, M.A.V.: Indexing moving objects using short-lived throwaway indexes. In: Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases (2009)
6. Feng, J., Lu, J., Zhu, Y., Mukai, N., Watanabe, T.: Indexing of Moving Objects on Road Network Using Composite Structure. In: Knowledge-Based Intelligent Information and Engineering Systems (2007)
7. Feng, J., Lu, J., Zhu, Y., Watanabe, T.: Index method for tracking network-constrained moving objects. In: Proceedings of the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Part II (2008)
8. Guohui, L., Yanhong, L., Jianjun, L., Shu, L., Fumin, Y.: Continuous reverse k nearest neighbor monitoring on moving objects in road networks. Inf. Syst. (2010)
9. Jensen, C.S., Lin, D., Ooi, B.C.: Query and update efficient b+-tree based indexing of moving objects. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, vol. 30 (2004)
10. Kang, H.-Y., Kim, J.-S., Li, K.-J.: Indexing moving objects on road networks in p2p and broadcasting environments. In: Carswell, J.D., Tezuka, T. (eds.) W2GIS 2006. LNCS, vol. 4295, pp. 227–236. Springer, Heidelberg (2006)

11. Kim, K.-S., Kim, S.-W., Kim, T.-W., Li, K.-J.: Fast indexing and updating method for moving objects on road networks. In: Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops (2003)
12. Kwon, D., Lee, S., Lee, S.: Indexing the current positions of moving objects using the lazy update r-tree. In: Proceedings of the Third International Conference on Mobile Data Management (2002)
13. Lai, C., Wang, L., Chen, J., Meng, X., Zeitouni, K.: Effective density queries for moving objects in road networks. In: Proceedings of the Joint 9th Asia-Pacific Web and 8th International Conference on Web-age Information Management Conference on Advances in Data and Web Management (2007)
14. Mouratidis, K., Yiu, M.L., Papadias, D., Mamoulis, N.: Continuous nearest neighbor monitoring in road networks. In: Proceedings of the 32nd International Conference on Very Large Data Bases (2006)
15. Patel, J.M., Chen, Y., Prasad Chakka, V.: Stripes: an efficient index for predicted trajectories. In: Proceedings of ACM SIGMOD International Conference on Management of Data (2004)
16. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel approaches to the indexing of moving object trajectories (2000)
17. Qin, L., Yu, J.X., Ding, B., Ishikawa, Y.: Monitoring aggregate k-nn objects in road networks. In: Proceedings of the 20th International Conference on Scientific and Statistical Database Management (2008)
18. Saltenis, S., Jensen, C.S.: Indexing of Moving Objects for Location-based Services. In: Proceedings of 18th International Conference on Data Engineering (2002)
19. Shahabi, C., Kolahdouzan, M.R., Sharifzadeh, M.: A road network embedding technique for k-nearest neighbor search in moving object databases. In: Proceedings of ACM International Symposium on Advances in Geographic Information Systems (2002)
20. Silva, Y.N., Xiong, X., Aref, W.G.: The rum-tree: supporting frequent updates in r-trees using memos. The VLDB Journal (2009)
21. Sun, H.-L., Jiang, C., Liu, J.-L., Sun, L.: Continuous reverse nearest neighbor queries on moving objects in road networks. In: Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management (2008)
22. Tao, Y., Papadias, D.: Mv3r-tree: A spatio-temporal access method for timestamp and interval queries. In: Proceedings of the 27th International Conference on Very Large Data Bases (2001)
23. Tao, Y., Papadias, D., Sun, J.: The TPR*-tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29 (2003)
24. Šaltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the Positions of Continuously Moving Objects. SIGMOD Record (2000)
25. Šidlauskas, D., Šaltenis, S., Christiansen, C.W., Johansen, J.M., Šaulys, D.: Trees or grids?: indexing moving objects in main memory. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (2009)
26. Wang, H., Zimmermann, R.: Snapshot location-based query processing on moving objects in road networks. In: Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (2008)
27. 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)
28. Yiu, M.L., Tao, Y., Mamoulis, N.: The bdual-tree: indexing moving objects by space filling curves in the dual space. The VLDB Journal (2008)

Real-Time Capable Data Management Architecture for Database-Driven 3D Simulation Systems

Jürgen Roßmann, Michael Schluse, Ralf Waspe, and Martin Hoppen

Institute for Man-Machine Interaction,
RWTH Aachen University,
Ahornstr. 55, 52074 Aachen, Germany
{rossmann,schluse,waspe,hoppen}@mmi.rwth-aachen.de
http://www.mmi.rwth-aachen.de

Abstract. State of the art 3D simulation applications like virtual testbeds for space robotics, industrial automation or even forest inventory require a highly flexible but still real-time capable data management system. For this, we combine a high-performance internal simulation database with external object-oriented databases into a new real-time capable data management architecture for database-driven 3D simulation systems.

To achieve this, we apply well-known database techniques to a 3D simulation system's internal object-oriented data management. Such a simulation database can dynamically adopt completely new data schemata, even at runtime. New simulation applications can then be designed by putting a domain specific schema and the corresponding data into an otherwise "empty" simulation database. To seamlessly combine the two databases we use a flexible interface that synchronizes schema and data.

Keywords: 3D Simulation System, Simulation Database, Event-Driven Graph Database, Object-Oriented Database, Database Synchronization.

1 Introduction

In this paper we introduce new methods to apply techniques common to databases to a 3D simulation system to simplify the realization of specialized database-driven simulation applications like virtual testbeds for space robotics, industrial automation or even forest inventory. Therefore, we developed a high-performance runtime simulation database, that can dynamically adopt new data schemata just like common databases can, but still meets the strong real-time requirements of simulation applications. Furthermore, we propose a flexible new way to link this real-time database with external object-oriented databases to combine their advantages in a new real-time capable data management architecture for database-driven 3D simulation systems.

The simulation database can be seen as an abstraction layer of the external database, hiding away the specific DBMS from the simulation system and its

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 262–269, 2011.

user. It furthermore works as a real-time capable "intelligent" cache for the virtually unlimited storage provided by an external database. Vice versa, based on the interface's read-write access the external database serves as a persistence layer for the simulation system, that additionally provides modern database features like access rights management and versioning. Attaching more than one simulation client, an active external database can even be used as a central data storage and communication hub for collaborative data access and distributed simulation applications, where changes by one client are actively pushed to all other participating clients.

2 Related Work

Though many attempts have been made to incorporate databases into 3D simulation systems and the like, no one to our knowledge has used them to their full extent. In the simplest scenarios, databases have been used to store additional information (meta information, documents, films ...) on scene objects ([7], [8]) or – as a next step – to store object positions etc. ([6]). Versioning is only supported in CVS-like systems ([12], [13]) taking place only on file level. The more sophisticated systems use the database to store the scene data itself, where some do support collaboration ([3], [4], [5], [6], [9]) while others do not ([1], [2], [11]). The support for different data schemata is not widespread among these systems. The simplest realizations allow schema alteration by adding attributes to generic base objects (e.g. [2]). The more advanced systems support different static ([10]) or dynamic ([11]) schemata.

3 Simulation Database

For our developments we are using the 3D simulation system VEROSIM®. An important aspect of this system is its modeling concept. With its object-oriented data model, objects are provided "with a meaning" according to the real world. The system is based on an event driven and object-oriented graph database called VSD (VEROSIM Active Simulation Database), which describes the components as well as their behavior and provides methods for parallel and distributed computing and visualization. Nodes not only provide data encapsulation but also mechanisms for interaction with the simulation environment implementing their behavior – this is what distinguishes this database from standard scene graphs.

All active components of the database are derived from an "Instance" class. Simulation data as well as the topology of the database is stored in so called properties. In this case "active" means that whenever a property is changed a signal is sent to all registered listeners. Furthermore, the container of the database itself is derived from the instance class and keeps track of all elements in the database in a set of indexed lists sorted by class type. Adding or removing instances from the database and thus from the index lists triggers a signal as well. Reflection is provided by so called meta-instances describing classes of

instances (name, inheritance, etc.) and meta-properties describing their properties (name, type, multiplicity, etc.). This enables the simulation system to query meta-instances and meta-properties by name. Dynamic schema generation allows for the creation of new meta-instances and meta-properties at run-time. This is used to enable the synchronization process described in the next chapters. The (simplified) databases of two exemplary simulation applications are shown in Fig. 1.

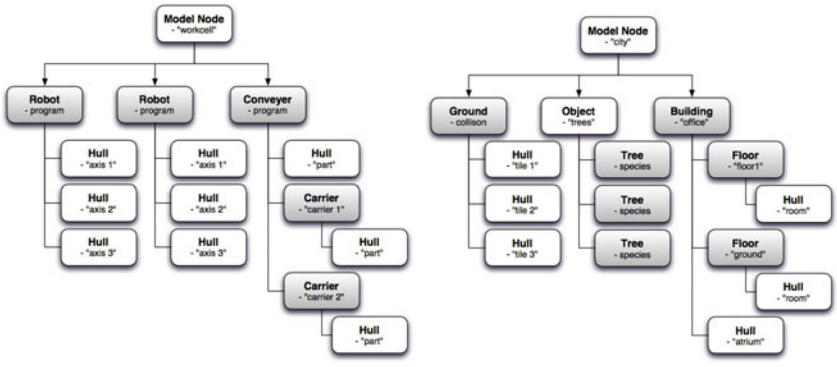


Fig. 1. Simplified graph database for a small robotic work cell (left) and a city model (right)

4 External Database

We now attach this internal database to an external data management system. For this, we are currently using the GML database management system SGJ [15]. This object-oriented geo DBMS offers instantiation of schemata using XML Schema Definition (XSD). This allows for the definition of arbitrary classes, inheritance hierarchies, relations and attributes. SGJ can be controlled using its proprietary Java API with a kernel based architecture. Alternatively it can be accessed using a transactional Web Feature Service (WFS-T).

Although until now our interface has only been tested using SGJ, the basic design is independent of the actual chosen DBMS. Given the flexibility of our data management architecture any object-oriented DBMS providing an adequate interface (e.g. with reflection) could be used.

5 Synchronization Mechanism

To seamlessly integrate the internal and external databases, we designed an interface between them providing a dynamic two-level synchronization. By replicating objects from the external database to the simulation database and keeping them "in sync" not only object data is synchronized. The data schema of the

external database is also replicated and thereby instantiated in the internal database making both "speak the same language" and providing a common data schema for the whole system. This way the system shows great flexibility towards changes in the data and even schema itself.

By using (replicated) objects instead of directly using database objects, details of the actually used database are hidden from all internal components of the simulation system (particles, physics, rendering, etc.) allowing for transparent real-time data access. The internal simulation database works as a cache to speed up repeating access patterns, which would otherwise lead to repetitive queries to the external database. Additionally, using an external database provides the simulation system with persistence as changes made to replicated objects can be resynchronized back into the database.

5.1 Schema Synchronization

The result of the schema synchronization – which is done only once during runtime initialization – is a schema mapping: classes are mapped to meta-instances and attributes (or relations) to meta-properties. This can be realized in two (combinable) ways.

In **schema matching**, a previously defined schema for the internal database is matched against the same schema in the external database (e.g. class *Building* matches meta-instance *Building*, attribute *storeys* matches meta-property *storeys* ...). Technically the internal schema is hard-coded and defined at compile-time. The matching process itself is name- and type-based using VSD's reflection API mentioned above. Schema matching is mainly used for real-time critical and built-in functionality like geometry representation for rendering.

In contrast, **schema transfer** is entirely performed at run-time using dynamic schema generation as described above. The database's schema is evaluated and meta-instances are defined without previously being hard-coded. Based on the class's attributes and relations, the appropriate meta-properties are created, defining data and structure of the schema within the simulation system.

5.2 Data Synchronization

After synchronizing the schemas data is synchronized on the object-level. No component of the simulation system has direct access to the external database. Instead, these components access data by means of local copies in the internal database, i.e. replicated versions of database objects. This built-in transparency provides for the flexibility, because any external database can easily be replaced by a different one. Furthermore, the replicated objects built up the cache functionality.

Object loading implies that an object from the database is replicated in the simulation database. This is realized by querying the object and instantiating an instance of the corresponding meta-instance (known from the schema mapping). Then all attributes and relations are copied to the corresponding properties and a mapping between the database's object and the replicated instance is inserted into the object mapping (Fig. 2).

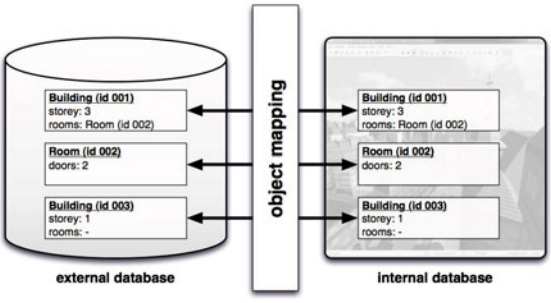


Fig. 2. Result of data synchronization: object mapping for city data

At any time, changes to the simulation database are automatically tracked. This includes changes to properties of instances, as well as the creation and deletion of instances. For this, the database interface registers itself to receive change notifications from the simulation database. Recognized changes are queued as internal transactions and can be written back to the database to resynchronize the internal and external state of the data. Depending on the application, this resynchronization is carried out immediately or by user request. Resynchronization implements a persistence layer for the simulation system as changes to the virtual environment can be made permanent. Thus simulation results can be saved and virtual environments can be manipulated permanently, right within the simulation system giving it more flexibility in the choice of possible applications (see section 7). Besides loading, instances can also be unloaded, provided that no transactions have been stored for them. Unloading is simply realized by deleting the instance from the internal database and removing the object mapping. Usually, unloading is used for dynamically streaming parts of large data sets, e.g. forest environments or city models.

6 Current Work

Currently, we are investigating the integration of change notifications from the external database as an aspect of data synchronization, a precondition for multi client synchronization. Therefore an active database is needed. We describe this scenario as a two-tiered architecture with multiple simulation systems connected to a central active database. This not only enables multi client operation on a consistent data set but also allows for the use of the database as a central communication hub and central data storage [16] that defines a common data schema by synchronizing it to all connected simulation systems.

In such a distributed simulation system, changes of an object are written back (i.e. are resynchronized) to the central database. All other connected simulation systems are informed about these changes and apply them to their local copies of the same object accordingly. Fig. 3 shows an example of such a communication sequence. A door is opened in the virtual environment on client 1, changing

the object's property open to true. This change is resynchronized to the central database where the object's property is set to true. This change is notified to all connected clients, including client 2 which sets the door's property accordingly, thus resynchronizing the system as a whole.

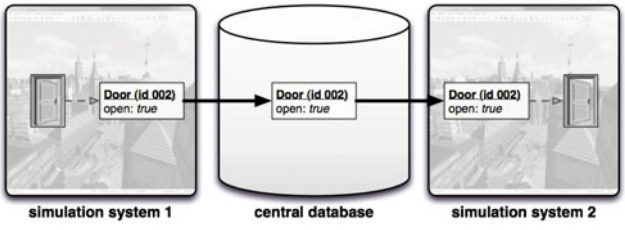


Fig. 3. Example for a communication sequence via the central database

To avoid notification loops, changes adopted from the central database are not interpreted in the same way as changes from within the local simulation system. Thus, after simulation system 2 sets the door's open property to true it will not resynchronize this change to the central database.

7 Applications

Several database-driven simulation applications using the data management architecture introduced in this paper have been realized so far. While they belong to very different fields of application they all benefit from the flexibility provided by our approach. One major application is the Virtual Forest[1], a science project [17] whose goal is to provide new means for forest inventory, serving as the basis for the development of decision support systems and the application of industrial automation techniques to the forest industry. Here, the proposed architecture has not only been used for simulations but also for an integrated tool for forest inventory (Fig. 4 right).

Another field of application are 3D city simulations. Here, SGJ databases with CityGML [14] models of cities like Stuttgart, Düsseldorf or Barkenberg (Kreis Recklinghausen) (Fig. 4 left) have been connected to VEROSIM. Simulations include driving a virtual car (with physically correct behavior) or flying a virtual helicopter through a city. The same was done using a dataset in the SEDRIS data format [18].

[1] The Virtual Forest project is co-financed by the European Union and the federal state of North Rhine-Westphalia, European Regional Development Fund (ERDF). Europe - Investing in our future.

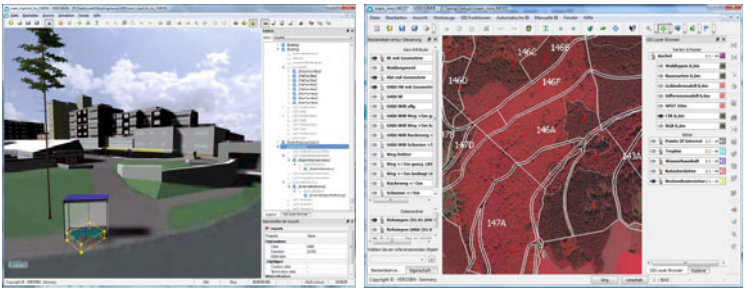


Fig. 4. A city model (left) and the inventory tool of the Virtual Forest (right)

The major goal of the research project FastMap[2] is the automatic generation of navigation maps as a basis for self-localization and navigation of mobile robots during the exploration of planetary surfaces. The introduced database interface is used to connect a Virtual Testbed and other data generators to a central database, thus using it as a common data store and communication framework.

8 Conclusion

Our concept of a real-time capable data management architecture for a 3D simulation system has already proven its great flexibility for different fields of application as seen in the previous section 7. It combines the advantages of well known database management techniques and state of the art 3D simulation technology to realize new applications within the new field of database-driven simulation. In the near future, it will build the basis for even more applications, for example in the context of product design and manufacturing. Using AutomationML [19], a standards-based approach to automation technology can be realized. The modeling concept of the language can be used in the different engineering processes (simulation, design . . .) within our system.

References

1. Takemura, H., Kitamura, Y., Ohya, J., Kishino, F.: Distributed Processing Architecture for Virtual Space Teleconferencing. In: Proc. of ICAT 1993, pp. 27–32 (1993)
2. Van Maren, G., Germs, R., Jansen, F.: Integrating 3D-GIS and Virtual Reality. Design and implementation of the Karma VI system. In: 10th Colloquium of the Spatial Information Research Center (1998)

[2] Gefördert von der Raumfahrt-Agentur des Deutschen Zentrums für Luft- und Raumfahrt e.V. mit Mitteln des Bundesministeriums für Wirtschaft und Technologie aufgrund eines Beschlusses des Deutschen Bundestages unter dem Förderkennzeichen 50 RA 1034.

3. Julier, S., Baillot, Y., Lanzagorta, M., Brown, D., Rosenblum, L.: BARS: Battlefield Augmented Reality System. In: NATO Symposium on Information Processing Techniques for Military Systems, pp. 9–11 (2000)
4. Masunaga, Y., Watanabe, C.: Design and Implementation of a Multi-modal User Interface of the Virtual World Database System (VWDB). In: Proceedings of the 7th International Conference on Database Systems for Advanced Applications (DASFAA 2001), pp. 294–301. IEEE Computer Society, Washington (2001)
5. Watanabe, C., Masunaga, Y.: VWDB2: A Network Virtual Reality System with a Database Function for a Shared Work Environment. In: Proceeding (367) Information Systems and Databases (2002)
6. Manoharan, T., Taylor, H., Gardiner, P.: A collaborative analysis tool for visualisation and interaction with spatial data. In: Proceedings of the Seventh International Conference on 3D Web Technology (Web3D 2002), pp. 75–83. ACM, New York (2002)
7. Borgatti, C., Felicori, M., Mauri, M.A., Calori, L., Guidazzoli, A., Pescarin, S., Diamanti, T., Liguori, M.C., Valentini, L.: Databases and virtual environments: a good match for communicating complex cultural sites. In: Longson, T. (ed.) ACM SIGGRAPH 2004 Educators Program (SIGGRAPH 2004). ACM, New York (2004)
8. Damer, B., Gold, S., Rasmussen, D., Neilson, M., Newman, P., Norkus, R., Bertelshems, B., Clancey, W.J., Sierhuis, M., Van Hoof, R., Shirley, M., Cochrane, T.: Data-Driven Virtual Environment Assembly and Operation: an extended abstract for the Virtual Ironbird Workshop. In: VIB Workshop Report, NASA Ames Research Center, Naval Postgraduate School (2004)
9. Kaku, K., Minami, H., Tomii, T., Nasu, H.: Proposal of Virtual Space Browser Enables Retrieval and Action with Semantics which is Shared by Multi Users. In: 21st International Conference on Data Engineering Workshops, April 05-08, p. 1259 (2005)
10. Haist, J., Coors, V.: The W3DS-Interface of Cityserver3D. In: Kolbe, G. (ed.) European Spatial Data Research (EuroSDR) u.a.: Next Generation 3D City Models. Workshop Papers: Participant's Edition, pp. 63–67 (2005)
11. Schmalstieg, D., Schall, G., Wagner, D., Barakonyi, I., Reitmayr, G., Newman, J., Ledermann, F.: Managing Complex Augmented Reality Models. In: IEEE Computer Graphics and Applications, pp. 48–57 (2007)
12. ENOVIA V6 Whitepaper, http://www.3ds.com/fileadmin/PRODUCTS/ENOVIA/PDF/WHITE-PAPERS/PCSWhitepaper-0807_final_July_29.pdf
13. AUTODESK® VAULT Whitepaper, http://images.autodesk.com/adsk/files/best_practices1.pdf
14. CityGML, http://www.citygml.org
15. CPA Systems GmbH, http://www.cpa-systems.de
16. Freund, E., Müller, M., Roßmann, J.: Data storage and flow control in automation systems by means of an active database. In: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 1999), pp. 235–240 (1999) ISBN 90 5199 475 3
17. Roßmann, J., Schluse, M., Bücken, A.: The Virtual Forest – Space and Robotics technology for the efficient and environmentally compatible growth-planning and mobilization of wood resources. In: FORMEC 2008, vol. 41, pp. 3–12 (2008) ISBN 978-3-9811335-2-3
18. SEDRIS Technologies, http://www.sedris.org
19. AutomationML, http://www.automationml.org

Collecting and Managing Network-Matched Trajectories of Moving Objects in Databases

Zhiming Ding[1] and Ke Deng[2]

[1] Institute of Software, Chinese Academy of Sciences
South-Fourth-Street 4, Zhong-Guan-Cun, Beijing 100190, P.R. China
[2] School of Information Technology & Electrical Engineering
The University of Queensland, Brisbane, QLD 4072 Australia
zhiming@iscas.ac.cn, k.deng@uq.edu.au

Abstract. To track network-matched trajectories of moving objects is important in a lot of applications such as trajectory-based traffic-flow analysis and trajectory data mining. However, current network-based location tracking methods for moving objects need digital maps installed at the moving object side, which is not realistic in a lot of circumstances. In this paper, we propose a new moving objects database framework, *Euclidean-batch-sampling and Network-matched-trajectory based Moving Objects Database (EuNetMOD)* model, to support network-matched trajectory tracking without digital maps installed at the moving object side.

Keywords: Database, Spatial temporal, Moving Object, Trajectory, Tracking.

1 Introduction

In recent years, the management of moving objects has been intensely investigated. Earlier work on MOD is mainly focused on Euclidean based solutions [1-2]. Euclidean-based trajectories are imprecise in describing the network paths the moving objects have taken, because multiple paths over the network can coexist between two consecutive sampling points.

More recently, increasing research interests are focused on network-constrained moving objects, with many effective models and algorithms proposed [3-8]. However, trajectories of network-constrained moving objects are not necessarily network-matched (see Figure 1). By "network-matched" we mean that the trajectory can be explained to only one path over the traffic network without ambiguity. In other words, the path between any two consecutive motion vectors of the trajectory is explicitly expressed by the two motion vectors (or by other ways) without having to call shortest path calculations. To ensure trajectories to be network-matched is very important in a lot of applications such as high-accuracy location tracking, trajectory-based traffic-flow statistical analysis, and trajectory-based data mining.

There exist several location tracking mechanisms for network-constrained moving objects [4-5, 8], which can track network-matched trajectories. However, they need moving objects to equip with digital maps (or "mobile maps") and to trigger location

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 270–279, 2011.

updates promptly when the moving objects transfer from one route to another. The installation of digital maps at the moving object side can greatly increase the cost of the whole system, and can also cause extra costs when the maps need to be refreshed.

In this paper, we propose a new moving objects database framework, **Eu**clidean-batch-sampling and **Net**work-matched-trajectory based **M**oving **O**bjects **D**atabase (EuNetMOD) model, which is a "mobile-map-free" tracking method. The main motivation of EuNetMOD is to support the tracking and expressing of network-matched trajectories at the server side without digital maps installed at the moving object side. This feature is quite useful – For example, most GPS-equipped taxis in China are with light-weighted GPS platforms (typically based on MCU, with no digital map, with very limited CPU, RAM, and storage). It would be greatly desirable if we can make use of these existing platforms in tracking network-matched trajectories without major changes.

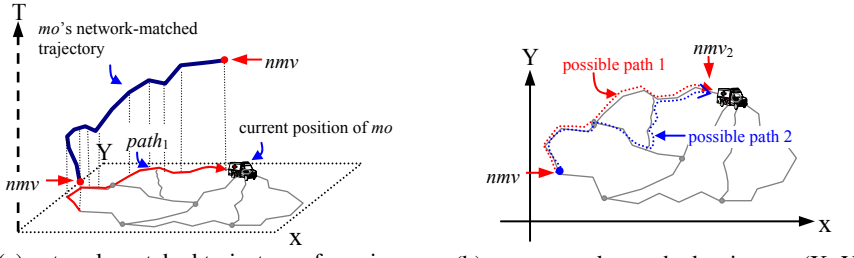
2 Overview of the EuNetMOD Mechanism

In EuNetMOD, moving objects can read motion vectors at any time and send them to the database server repeatedly. We call the operations of "reading motion vectors" and "sending them to server" as "sampling" and "location update" respectively.

For a certain moving object mo, its locations can be tracked by using a "Fixed-Time Sampling plus Fixed-Time Location Update (FTS + FTLU)" method. That is, in every τ_s time (say in every 15 seconds), mo samples a Euclidean-based motion vector of the form (t, (x, y), v, d), where t is a time stamp, and (x, y), v, d are the location, the speed, and the direction of mo at time t respectively. Besides, every time when mo's direction change or speed change exceeds certain predefined thresholds, mo will trigger an extra sampling. The sampled motion vectors are temporarily kept in the local storage of the moving object and in every τ_u time (say in every 3 minutes), mo sends the sampled motion vectors to the database server in batch.

Except the above mentioned "FTS + FTLU" location tracking strategy, we can also adopt other location update policies such as "FDS + FTLU" (Fixed-Distance Sampling plus Fixed-Time Location Update) or "FDS + FDLU" (Fixed Distance Sampling plus Fixed-Distance Location Update). The general rule is that mo should sample its Euclidean based motion vectors relatively densely and sends the sampled data in batch to the server in relatively sparse time intervals.

When the database server receives a location update message which contains multiple Euclidean-based motion vectors, it will match the Euclidean-based motion vectors to the network so that we can get network-matched motion vectors of the form (*t,* (*x, y*)*, v, d, npos*), where *t,* (*x, y*)*, v,* and *d* are from the original Euclidean-based motion vector and *npos* is the corresponding network position. After that, it will find a network path between any two neighbouring network-matched motion vectors so that the network path for the whole trajectory becomes available. Then it will discard unimportant motion vectors which are implicitly inferable from its predecessor and successor. The resulted trajectory is "network-matched" since the path information is explicitly expressed in the trajectory, as shown in Figure 1(a). Figure 1(b) shows a non network-matched trajectory which can have ambiguity in deciding the path between two neighbouring sampling points.



(a) network-matched trajectory of moving object *mo*

(b) non network-matched trajectory (X×Y)

Fig. 1. Network-matched vs. non network-matched trajectories

As stated above, since the path information (expressed as an edge sequence) is contained in the trajectory, unimportant motion vectors can be discarded if it is inferable from its neighboring motion vectors. In extreme cases, a network-matched trajectory can have only two network-matched motion vectors nmv_1, nmv_2 and the path between them (if the moving object runs in roughly steady speed), as illustrated in Figure 1. Therefore, the storage of the network-matched trajectory can be optimized while the precision can still be reserved.

Each network-matched trajectory can describe a continuous movement of the moving object. To describe the dynamic locations of a moving object over a long period of time (say 3 months), multiple trajectories are needed. For simplicity, we still call the multiple trajectories of the same moving object as "trajectory" in this paper.

In real-world applications, there exist situations when moving objects run outside the traffic network occasionally. In this case, EuNetMOD will keep it in the trajectory as its original Euclidean form. Besides, EuNetMOD allows network-matching to have multiple candidate network-points coexisting so that the model can deal with more flexible situations as described in Section 4.

3 Modeling Network-Matched Trajectories of Moving Objects

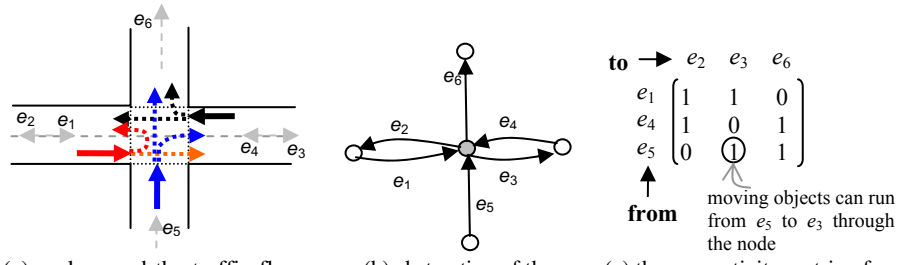
In this section, we describe how transportation networks and network-matched trajectories are represented in EuNetMOD. For simplicity, we assume that the transportation network is spatially embedded in the $X \times Y$ plane.

In EuNetMOD, we use an edge-based model to represent the transportation network. A network *Net* is modeled as the form $Net = (E, N)$, where E is a set of directed edges and N is a set of nodes.

A direct edge (or simply edge) $e \in E$ is defined as the form $e = (eid, geo, len, nid_s, nid_e)$, where $eid \in$ string is the identifier of e; $geo = (p_1, p_2, \ldots, p_n) \in$ polyline is the geometry of e where p_i $(1 \le i \le n) \in$ point is the ith vertex of the polyline; $len \in$ real is the length of e, and nid_s, $nid_e \in$ string are the identifiers of the starting and ending nodes which are connected by edge e.

A node $n \in N$ is defined as the form $n = (nid, loc, (eid_i)_{i=1}^{m}, mat)$, where $nid \in$ string is the identifier of n; $loc \in$ point is the location of n; eid_i $(1 \le i \le m) \in$ string is

the identifier of the *i*th edge connected by *n*; and *mat* is the connectivity matrix of *n*, which describes the traffic transferability between different edges through the node. Figure 2 shows a road structure, the corresponding network abstraction, and connectivity matrix respectively.



(a) node *n* and the traffic flows in it

(b) abstraction of the network

(c) the connectivity matrix of *n*

Fig. 2. A node and its connectivity matrix

As illustrated in Figure 2, the connectivity matrix *mat* contains possible matches of traffic flows in the edges connected by the node, and the element value associated with each match can assume either 0 or 1, which indicates whether moving objects can transfer between the corresponding edges through the node. For instance, moving objects can transfer from e_5 to e_3 through the node, and therefore the corresponding element value is 1.

A network point *netpoint* ∈ D(*Net*) is a point residing in network *Net*, where D(*Net*) =$E \times [0, 1] \cup N$ is the domain of network positions in *Net*. In general, *netpoint*'s position can have two possibilities. It can either reside in an edge, or reside in a node. In the former case, a real number $p \in [0, 1]$ has to be specified to further describe the position of *netpoint* inside the edge (for any edge, we suppose that its total length is 1, so that any location in the edge can be presented by $p \in [0, 1]$).

For a moving object *mo*, its Euclidean-based motion vector *emv* can be defined as the form *emv* = (*t*, (*x, y*), *v*, *d*) where *t* ∈ real is the time instant when the motion vector is sampled, (x, y) ∈ point, *v* ∈ real, and *d* ∈ real are the location, the speed, and the direction of the moving object at time *t* respectively.

For an Euclidean-based motion vector *emv*, its corresponding network-based motion vector *nmv* can be defined as: *nmv* = (*t*, (*x, y*), *v*, *d*, *nps*), where *t*, (*x, y*), *v* and *d* are from *emv*, and *nps* ⊆ D(*Net*) is the corresponding network points computed from *emv*. Normally, *nps* only contains one network point (that is, |*nps*| = 1). However, it can also contain multiple network points (|*nps*| > 1) to accommodate the situation when multiple candidate network matches of (x, y) coexist which can not be eliminated through the currently available information (we call *nmv* "multi-matched" in this case, see Subsection 4.2.1). On the other hand, if *emv* can not be matched to the network, then *nps* is set to ∅ (|*nps*| = 0), to accommodate the situation when the moving object runs outside the network.

The network-matched trajectory of *mo*, denoted as *nmtr*, is defined as the form *nmtr* = (nmv_1, $path_1$, nmv_2, $path_2$, …nmv_n), where nmv_i ($1 \le i \le n$) is the *i*th network-based

motion vector, and $path_i = (e_1, e_2, \ldots e_m)$ $(1 \le i \le m)$ is a sequence of edges describing the network path the moving object takes between nmv_i and nmv_{i+1}. $path_i$ can take the undefined value ($\perp$) if one of mv_i and mv_{i+1} or both of them are not network-matched or multi-matched.

4 Sampling and Generating Network-Matched Trajectories

In this section, we describe the location tracking mechanism used in EuNetMOD for sampling and generating network-matched trajectories of moving objects, which is called the *Front-end Euclidean Batch Sampling and Backend Network Matching* method. The main aim is to avoid using digital maps at the moving object side when tracking the network-matched trajectories at the server side. The basic idea is:

(1) the moving object samples Euclidean-based motion vectors densely (say in every 15 seconds or in every 200 meters, plus extra samplings according to speed and direction changes), keeps them at the local buffer temporarily, and sends the buffered motion vectors to the server as a location update message relatively sparsely (say in every 3 minutes or in every 2000 meters). A location update message *lumsg* is defined as the form: $lumsg = ((t_i,(x_i, y_i),v_i, d_i))_{i=1}^{n}$, where $(t_i,(x_i, y_i), v_i, d_i)$ is the ith motion vector the moving object has sampled. The first motion vector of *lumsg* can assume the undefined value ($\perp$) when the moving object is starting a new journey, and the last motion vector of *lumsg* can be undefined when it is finishing an old journey (see Algorithm 1). The undefined motion vectors help the server to break up the sequence of samplings into multiple trajectories with each trajectory describing a continuous movement of the moving object.

(2) after receiving a location update message *lumsg* which contains a sequence of Euclidean-based motion vectors, the database server will first match every motion vector to the network and get a network-matched location update message $nlumsg = ((t_i,(x_i, y_i), v_i, d_i, nps_i))_{i=1}^{n}$ (we call this procedure "network-matching"), then find a path over the network which can link the sequence of motion vectors (this procedure is called "path-finding"), and finally, discard unimportant motion vectors of the trajectory so that only key motion vectors for describing the shape of the trajectory is kept to save the storage (this procedure is called "trajectory-optimizing"). The resulted trajectory is saved in the database (we call the trajectory kept in the database "database trajectory"). For simplicity, we suppose that the wireless communication channel is reliable so that no location update message will get lost.

4.1 Sampling Euclidean-Based Motion Vectors from the Moving Object Side

In general, we can have multiple alternatives under the general rules of FEBS-BNM, for instance FTS+FTLU, FDS+FDLU, and FDS+FTLU, as stated in Section 2. Except the time or distance intervals, in FEBS-BNM we should also consider direction and speed changes during the sampling of motion vectors. The reason is that if the moving object changes its direction sharply, it is very probably transferring to another edge so that the sampling will not miss key points in the network. If the moving object changes speed dramatically, the moving pattern is changed and the changing point should also be recorded. In this way, more key motion vectors can be included in the trajectory to describe the movement of the moving object.

Algorithm 1 shows the FTS+FTLU location update algorithm running at the moving object side. Other location update policies can be implemented similarly.

Algorithm 1. FTS + FTLU Location Update Algorithm (running at moving object side)

General Arguments:

τ_s, τ_u ; //time intervals for motion vector sampling and for location updates
δ, ξ; //direction / speed change thresholds for triggering extra samplings

```
1.  lumsg=NULL;
2.  Append (lumsg, ⊥);
3.  currentTime = getCurrentTime();
4.  readGPS(x, y, v, d);
5.  emv = (currentTime, (x, y), v, d);
6.  Append (lumsg, emv);
7.  lastSamplingTime = lastUpdateTime = currentTime;
8.  WHILE (mo is running) DO
9.    currentTime = getCurrentTime();
10.   ReadGPS(x, y, v, d);
11.   IF ((currentTime – lastSamplingTime ≥ τs) ∨ (d – emv.d) ≥ δ) ∨ (v – emv.v) ≥ ξ)) THEN
12.     emv = (currentTime, (x, y), v, d);
13.     Append (lumsg, emv);
14.     lastSamplingTime = currentTime;
15.   ENDIF;
16.   IF (currentTime – lastUpdateTime ≥ τu) THEN
17.     SendToSVR(lumsg);
18.     lumsg=NULL;
19.     lastUpdateTime = currentTime;
20.   ENDIF;
21. ENDWHILE;
22. Append (lumsg, ⊥);
23. SendToSVR(lumsg).
```

In Algorithm 1, the function Append (*lumsg*, *mv*) appends a motion vector *mv* to a sequence *lumsg*. The function getCurrentTime() reads the current machine time. The function readGPS(*x*, *y*, *v*, *d*) reads GPS values and save them to the specified parameters. The function SendToSVR(*lumsg*) sends the location update message *lumsg* to the server.

4.2 Generating Network-Matched Trajectories at the Server Side

In subsections 4.2.1 through 4.2.3, we first assume that the database trajectory of the moving object is null so that the discussion can be simplified, and then in subsections 4.2.4 we present the general trajectory generating algorithm with non-null database trajectories considered.

To match the sampling points to the network, if we were not using two directed edges to express a route with dual directions (or "dual route"), we could certainly use the map-matching method proposed in [9] which depends on distance to compute the path. However, in EunetMOD, both the edges of the traffic network and the sampling points have directions and we should take directions into account to find the right network path. For instance, because a dual route can have two directions, it is expressed with two edges of opposite directions but of nearly the same geometry. As

a result, a sampling point can be matched to a wrong edge by the method of [9], which can further lead to a wrong path-matching. To overcome this problem, in EuNetMOD we take directions into account in map-matching as shown in Figure 3.

4.2.1 Matching Euclidean-Based Motion Vectors to the Traffic Network

Whenever the database server receives a location update message *lumsg* of the form $lumsg = ((t_i,(x_i, y_i), v_i, d_i))_{i=1}^{n}$ from a certain moving object whose identifier is *moid*, it will first conduct network-matching procedure to match the Euclidean-based motion vectors in *lumsg* to the traffic network and get a network-based location update message $nlumsg = ((t_i,(x_i, y_i), v_i, d_i, nps_i)_{i=1}^{n}$. To speed up the network-matching process, we need to use the memory-based R-tree index on the traffic network. We suppose the threshold for GPS measuring errors is ε_{gps}.

Suppose $emv_i = (t_i, (x_i, y_i), v_i, d_i)$ $(1 \le i \le n)$ is an arbitrary motion vector in *lumsg*. According to ε_{gps}, (x_i, y_i) can be matched to multiple candidate network points with each of them within the ε_{gps} distance to (x_i, y_i) and corresponding to an edge or a junction. In this case, the system will further check emv_i's direction and the edges' directions, sometimes also emv_i's predecessors and successors to eliminate inappropriate candidates. In most cases, we can have one and only candidate left as the result of the network-matching procedure, as shown in Figure 3.

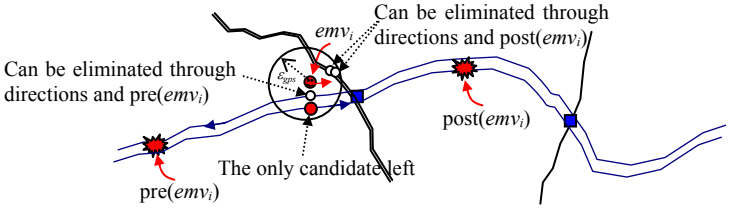


Fig. 3. Network-matching of Euclidean-based motion vectors

However, sometimes we can still have abnormities. For instance, even though all the currently available information is checked, we still have multiple candidates remained for emv_i, especially when emv_i is close to the end of *lumsg*. In this case, the multiple candidate network points will be kept in nps_i and when next location update message is received, further elimination will be conducted based on the newly received motion vectors, as described in Algorithm 2. Besides, there are cases when emv_i can not be matched to the network if the moving object is running outside the road network or if the map is not in detail enough. In this case, nps_i is simply set to $\varnothing$.

4.2.2 Path-Finding Based on Network-Matched Motion Vectors

Suppose that $nlumsg = ((t_i, (x_i, y_i), v_i, d_i, nps_i)_{i=1}^{n}$ is the network-based location update message resulted from the network-matching procedure. The server will then conduct the path-finding procedure so that for any two adjacent motion vectors in *nlumsg*, a network path can be found between them.

Let's consider two neighboring motion vectors $nmv_s = (t_s, (x_s, y_s), v_s, d_s, nps_s)$ and $nmv_e = (t_e, (x_e, y_e), v_e, d_e, nps_e)$. Suppose that $path_{se}$ is the network path between them. If $|nps_s| = 0$ or $|nps_e| = 0$, then at least one of them are not network-matched and in this case $path_{se}$ is set to null. If $|nps_s| > 1$ or $|nps_e| > 1$, then at least one of them is multi-matched and needs to be further processed so that $path_{se}$ is set to the undefined value ($\perp$) in this case.

If $|nps_s| = 1$ and $|nps_e| = 1$, then $path_{se}$ can be computed and there are several possibilities in this case, as shown in Figure 4 (suppose that np_s and np_e are the corresponding network points contained in nps_s and nps_e respectively, and path(np_s, np_s) is the complete path from nps_s to nps_e).

(1) If np_s and np_e are in the same edge or in the same node (see cases 1, 2), or if np_s and np_e are in adjacent edges or nodes (see cases 3, 4), then $path_{se}$ is set to null since the path information is already contained in the network-points and can be omitted;

(2) If np_s and np_e are not adjacent in the network (see case 5), then $path_{se}$ is set to the shortest path between them (the first and the last edges of the path can be omitted if they are contained in np_s and np_e already).

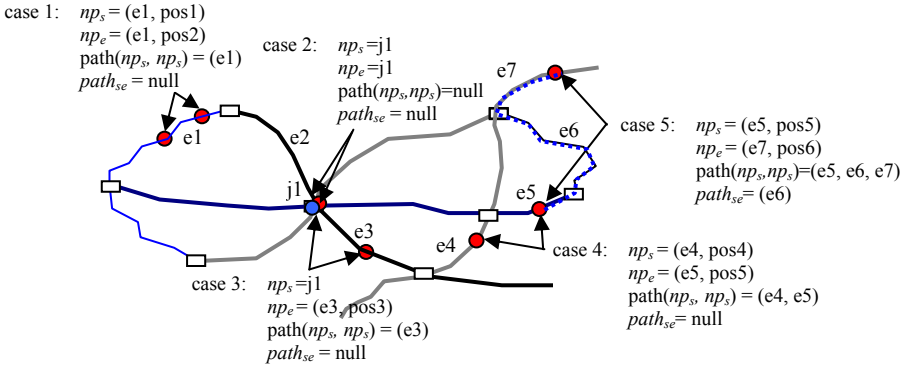


Fig. 4. Path-finding between adjacent motion vectors

From the above description we can see that the shortest path computation is still needed occasionally (see case 5 of Figure 4), which can cause errors since shortest path is not always the actual path. However, the shortest path computation is very rarely called if the sampling interval is short enough. The goal of EuNetMOD is to limit the shortest path computation to the minimum by adjusting the sampling time interval τ_s. Besides, in EuNetMOD, np_s and np_e are not far away because of dense samplings. In most cases the shortest path is the actual path the moving object takes.

4.2.3 Discarding Unimportant Motion Vectors and Appending the Resulted Network-Matched Trajectory to the Database Trajectory

After the path-finding is conducted, we can get a network-matched trajectory $nmtr^* = (nmv_1, path_1, nmv_2, path_2, \ldots, nmv_n)$, which contains all the sampling points received from the moving object plus the extra path information. Since the samplings are conducted densely, $nmtr^*$ may contain too many motion vectors so that it is very storage consuming. To reduce the size of the trajectory, the server needs to conduct

trajectory optimizing procedure to discard unimportant moving vectors as many as possible while reserving the shape of the trajectory.

During trajectory-optimizing, the system checks the motion vectors in $nmtr^*$ one by one (except the first and the last motion vectors). Suppose that $nmv_i = (t_i, (x_i, y_i), v_i, d_i, nps_i)$ $(1 < i < n)$ is an arbitrary motion vector in $nmtr^*$. nmv_i can be discarded if:

$$\text{distance}(\text{pos}(nmv_i), \text{pos}_{\text{cmp}}(nmv_{i-1}, nmv_{i+1}, t_i)) \leq \varepsilon$$

where $\text{pos}_{\text{cmp}}(nmv_{i-1}, nmv_{i+1}, t_i)$ is the computed position of the moving object at time t_i through interpolation based on nmv_{i-1} and nmv_{i+1}. In other words, if the spatio-temporal information of nmv_i can be inferred from its predecessor and successor (for instance, when the moving object runs with roughly steady speed between nmv_{i-1} and nmv_{i+1}), then nmv_i can be deleted without changing the shape of the trajectory curve. When deleting *nmv*, the related paths need to be merged so that the over all path information will not be lost.

After discarding of unimportant motion vectors, the resulted network-matched trajectory is saved to the server as the database trajectory of the moving object.

4.2.4 General Generating Algorithm for Network-Matched Trajectories

In Subsections 4.2.1 through 4.2.3 we assume that the database trajectory of the moving object is null. However, in most cases, the database trajectory is not null so that the processing becomes a little bit more complicated.

Suppose that the database trajectory of the moving object (whose identifier is *moid*) is $nmtrDB = (\underline{nmv}_1, \underline{path}_1, \underline{nmv}_2, \underline{path}_2, \ldots, \underline{nmv}_m)$ and the newly received location update message is $lumsg = ((t_i,(x_i, y_i), v_i, d_i))_{i=1}^{n}$. The server will first conduct network-matching on *lumsg*, and get a network-based location update message $nlumsg = ((t_i, (x_i, y_i), v_i, d_i, nps_i)_{i=1}^{n}$. After that, the server needs to concatenate *nmtrDB* and *nlumsg* into an overall trajectory $traj^*$, and then conduct path-finding and trajectory-optimizing based on $traj^*$. As stated earlier, there may exist some motion vectors remaining multi-matched in *nmtrDB* (usually close to the end of *nmtrDB*, with the associated paths undefined "⊥"). Therefore, the path-finding and trajectory-optimizing procedures should include these motion vectors as well as the newly received motion vectors. The general algorithm for generating network-matched trajectories at the server side is described in Algorithm 2.

Algorithm 2. Network-matched trajectory generation (running at the server side)

```
INPUT: moid;                                  //identifier of the moving object
       lumsg = (emv_i)_{i=1}^{n};             //location update message
1. nmtrDB = retriveTraj(moid);                //database trajectory
2. nlumsg = networkMatch(lumsg);
3. traj* = concat(nmtrDB, nlumsg);
4. nmtr* = pathFind (traj*);
5. nmtr* = trajOptimize(nmtr*)
6. writeDB (moid, nmtr*).
```

In Algorithm 2, the function retriveTraj(*moid*) retrieves the trajectory from the database according to the moving object identifier *moid*. The function

networkMatch(*lumsg*) matches Euclidean-based motion vectors to network. The function concat (*nmtrDB*, *nlumsg*) concatenates *nmtrDB* and *nlumsg* into a new trajectory *traj**. The functions pathFind(*traj**) and trajOptimize(*nmtr**) conduct path-finding and trajectory optimizing respectively. The function writeDB (*moid*, *nmtr**) writes *nmtr** into the database.

5 Performance Evaluation and Conclusion

To evaluate the performance of the EuNetMOD model, we have conducted a series of experiments based on the prototype system we have implemented. The experimental results show that by choosing appropriate sampling and location update time intervals, EuNetMOD can provide high network-matching precision with satisfactory query and storage performances.

Tracking and managing network-matched trajectories of moving objects is important in representing the precise dynamic locations of moving objects over the network. In this paper, we propose the EuNetMOD model, which can track and manage network-matched trajectories of moving objects. EuNetMOD does not need a digital map installed at the moving object side while the server side can still track precise network-matched trajectories so that it is flexible in real-world application.

Acknowledgements. The work was partially supported by NSFC under grand number 60970030.

References

[1] Wolfson, O., Yin, H.: Accuracy and Resource Consumption in Tracking and Location Prediction. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750. Springer, Heidelberg (2003)

[2] Güting, R.H., Böhlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., Vazirgiannis, M.: A Foundation for Representing and Querying Moving Objects. ACM Transactions on Database Systems 25(1), 1–42 (2000)

[3] Vazirgiannis, M., Wolfson, O.: A Spatiotemporal Model and Language for Moving Objects on Road Networks. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 20–35. Springer, Heidelberg (2001)

[4] Civilis, A., Jensen, C.S., Pakalnis, S.: Techniques for Efficient Road-Network-Based Tracking of Moving Objects. IEEE Trans. Knowl. Data Eng. 17(5) (2005)

[5] Tiesyte, D., Jensen, C.S.: Efficient Cost-Based Tracking of Scheduled Vehicle Journeys. In: Proc. of MDM 2008, Beijing, China (April 2008)

[6] Almeida, V., Güting, R.H.: Indexing the Trajectories of Moving Objects in Networks. GeoInformatica 9(1), 33–60 (2005)

[7] Güting, R.H., Almeida, V.T., Ding, Z.: Modeling and Querying Moving Objects in Networks. VLDB Journal 15(2), 165–190 (2006)

[8] Ding, Z., Zhou, X.: Location Update Strategies for Network-Constrained Moving Objects. In: Haritsa, J.R., Kotagiri, R., Pudi, V. (eds.) DASFAA 2008. LNCS, vol. 4947, pp. 644–652. Springer, Heidelberg (2008)

[9] Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On Map-Matching Vehicle Tracking Data. In: Proc. of VLDB 2005, Trondheim, Norway, pp. 863–864 (2005)

On Guaranteeing k-Anonymity in Location Databases

Anh Tuan Truong[1], Tran Khanh Dang[1,*] and Josef Küng[2]

[1] Faculty of Computer Science and Engineering, HCMC University of Technology, VNUHCM, Ho Chi Minh City, Vietnam
{anhtt,khanh}@cse.hcmut.edu.vn
[2] Institute for Application Oriented Knowledge Processing, Johannes Kepler University of Linz, Austria, Europe
josef.kueng@faw.jku.at

Abstract. The development of location-based services and mobile devices has lead to an increase in the location data. Through the data mining process, some valuable information can be discovered from location data. However, the attackers may also extract some private (sensitive) information of the user and this can make threats against the user location privacy. Therefore, location privacy protection becomes a key factor to the success in privacy preserving in location-based services. In this paper, we propose a new approach as well as an algorithm to guarantee k-anonymity in a location database. The algorithm will maintain the association rules which have significance for the data mining process. Moreover, the algorithm also considers excluding new significant association rules created during the run of the algorithm.

Keywords: k-anonymity, location databases, data mining, privacy protection.

1 Introduction

Today, advances in location technologies and wireless communication technologies enable the widespread development of location-based services (LBSs). When using services, the user may face with risks because the location information of the user can disclose some private information. Therefore, we need to protect the location information of the user from attacking of malefactors. In this paper, we will focus on protecting the user's location at time when the location data is stored in the database for data mining purposes. This paper will improve the approach which was proposed in [2, 9] and will use this improved approach to anonymize the location database to achieve an effective k-anonymous version. This approach does not use two popular techniques (generalization and suppression) because data after anonymizing by these techniques may not be significant to the data mining processes. The approach will use a technique which is called *Migrate Member technique* to anonymize the database [2]. The approach also considers the result of data mining process by maintaining association rules that are significant to the data mining process.

* Part of this research had been done as the author was at FAW institute, JKU, Linz, Austria as a visiting researcher.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 280–287, 2011.

2 k-Anonymity Techniques, M3AR Algorithm and Problems

In [2], the authors proposed the *Migrate Member technique* (and a variant in [9]) to anonymize the database to achieve a k-anonymous version. The technique first groups tuples of original data into separate groups by the similarity in quasi-identifier values. Then, the groups, which have less than k tuples, will be transformed into the satisfied ones by performing some Migrate Member operations. A satisfied group will have at least k tuples in it. The database achieves a k-anonymity version if all groups must be satisfied after the processing. The authors also proposed an algorithm called M3AR (Migrate Member Maintenance Association Rules) to concretize the approach.

With M3AR, they guarantee k-anonymity for the database while still maintaining the significant association rules. However, it remains many unsatisfied groups, which the algorithm can not transform them into the satisfied ones, after processing. Therefore, the algorithm may need more time and pay the "cost" to anonymize these unsatisfied groups. The cause of this is that M3AR selects a random unsatisfied group for each process step and thus this group may not be good for this step. As a result, this group can receive more tuples than its need, thus we may have no tuples to anonymize other groups. Moreover, M3AR did not also consider reducing new significant association rules that are generated during the process. Because these new significant association rules can interfere in the input data of the data mining process, it can make the result of the data mining process less valuable.

In next sections, we will propose some solutions to solve the problems of the algorithm M3AR. We also propose a new algorithm to anonymize the location database to achieve an effective k-anonymous version. Moreover, the algorithm also reduces new significant association rules generated during the run of the algorithm.

3 Definitions and Values Calculation for Proposed Algorithm

As in [3], *Quasi-Identifiers* (QI) is the set of attributes whose values may be used, possibly together with external information, to re-identify the user's data. For the location database, we will consider the *QI* (refer [3] for details) will include a location attribute and a time attribute. For simplification, we will only consider the location attribute in this paper. The time attribute will leave as future works.

3.1 Definitions

This section will give some definitions which will be used in the algorithm:

Definition: A group is a set of tuples. Moreover, all tuples in a group must have the same *QI* values. A group satisfies k-anonymity if it has at least k tuples or has no tuples in it. Otherwise, we call this group as an unsatisfied group.

Normally, the data mining process will consider association rules which occur frequently in the database. Therefore, the algorithm should try to retain these rules. We call these rules as significant rules. In the algorithm, two thresholds (*t_s* and *t_c*) will be provided to specify whether an association rule is significant or not. An association rule is significant if its support value is greater than *t_s* and its confidence value is also greater than *t_c*. Conversely, the association rule is insignificant.

Definition: A change between two groups $a \rightarrow b$, where a and b are groups, will change all *QI* values of some tuples in a to the correlative values in b. For example, group a has two tuples with *QI* is *(x1, y1, t1)* and group b has three tuples with *QI* is *(x2, y2, t2)*, the change $a \rightarrow b$ will form group b which has five tuples. The additional tuples are from group a and their *QI* attributes are changed to *(x2, y2, t2)*.

3.2 Values Calculation

With our algorithm, we will try to transform unsatisfied groups into satisfied ones. To do this, the algorithm will find the changes which will be performed to transform these unsatisfied groups to satisfied groups. Moreover, the algorithm also maintains significant association rules of the database. Thus, the algorithm should find the suitable changes in order that when performing these changes, these significant association rules will not be lost. From [2], we will calculate the maximal number of tuples which we can alter so that the association rule is still significant: We have a significant association rule $A \rightarrow B$, s is the support value and c is the confident value of this rule, *total* is the number of tuples in the database. We have two cases:

- A is changed, we have:

$$n = MIN\left(total * (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c * (1 - t\_c)} \right\rfloor\right) \quad (1)$$

- B is changed, we have:

$$n = MIN\left(total * (s - t\_s), \left\lfloor \frac{s * total * (c - t\_c)}{c} \right\rfloor\right) \quad (2)$$

We also have the case: both A and B will be changed. However, we notice that this case is similar to the case 1: A is changed (see [2] for details). As discussed above, the algorithm also guarantees that no new significant rule will be generated because the new significant rules may affect the result of the data mining process. Therefore, we also calculate the maximum number of tuples which we can add to a rule without generating new significant association rules. We will consider following cases: A will be added; B will be added and both A and B will be added. Only the last case may be make an insignificant rule become a significant one. Therefore, we have:

$$n = MIN\left(total * (t\_s - s), \left\lfloor \frac{s * total * (t\_c - c)}{c * (1 - t\_c)} \right\rfloor\right) \quad (3)$$

Where n is the maximal number of tuples, which can be added. *total* is the number of tuples in the database. The algorithm will use these maximal numbers to calculate cost for each change. The cost of a change will be mentioned in next sections.

4 Proposed Algorithm

Clearly, the objectives of the proposed algorithm are to perform the changes to transform unsatisfied groups into satisfied ones, and to maintain the significant association rules. Moreover, the algorithm should also reduce the number of new significant association rules that are created while running the algorithm. We call the

maintaining significant association rules and reducing the number of new significant association rules as proposed algorithm's goals.

The algorithm should guarantee that the goals will not be violated when the changes are performed. For each unsatisfied group, the algorithm will choose a/some group(s), which is the other unsatisfied group or the satisfied group, to form the changes. However, the algorithm does not choose these groups randomly; it will choose the best "compatible" groups so that when performing the changes between the unsatisfied group and these "compatible" groups, they have the least effect on the algorithm's goals. To do this, the algorithm will calculate "cost" for each change. Then it will choose the changes which have the least cost. While seeking these best "compatible" groups, the algorithm should concern the following issues: (i) Consider two-way for the changes between two groups; (ii) Choose the changes which have the least effect on the goal; (iii) A group can receive or distribute tuples more than one time; (iv) A group can receive tuples from different groups; (v) Prioritize the combination of two unsatisfied groups when we have some combinations that have same cost; (vi) For unsatisfied groups, prioritize the receipt of tuples from satisfied groups and the distribution of tuples to another unsatisfied groups.

Moreover, as discussed above, the algorithm should assign a priority degree for each unsatisfied group in order to determine which groups will be processed first. In the previous papers [2], their algorithm chose the current transformed unsatisfied group randomly. Therefore, this group may receive all of tuples that are available for distribution and we will not have enough tuples for other unsatisfied groups. As a result, we may get more unsatisfied groups after finishing the algorithm. In the algorithm, we will try to transform many more unsatisfied groups into the satisfied ones by assigning a priority degree for each unsatisfied group. To assign the priority degree for unsatisfied groups, the algorithm will base on criteria:

- Prioritize unsatisfied groups in which the number of tuples is closer to k: Clearly, unsatisfied groups, which the number of its tuples is closer to k, will be transformmed to the satisfied ones more easily.
- Prioritize unsatisfied groups which can not distribute tuples.

The algorithm will try to finish the anonymization of current unsatisfied group before working with next unsatisfied groups. An unsatisfied group can be transformed into a satisfied one if one of two following cases can be performed without affecting the goals: (i) all its tuples are distributed to other groups; (ii) it adds some tuples from other groups so that the number of its tuples is greater than k. In the second case, if a great number of tuples can be added to current unsatisfied group without affecting the goals, the group should only add enough tuples. It means that the number of group's tuples after processing should be equal to k. The remaining tuples will be left for other unsatisfied groups which are processed later.

In this paper, we also apply the grid based solution [1] to the location attribute of the location database to reduce the number of maintained association rules. As a result, the algorithm will run more quickly. The idea of this solution is that the exact location values will be anonymized into grid cells. With this solution, the algorithm will create a grid which covers the space containing the locations of the users in the database. After that, the locations of the users will be anonymized into this grid's cell.

The algorithm can be described as following pseudo code:

Name: *k_anoymization()*

Input: Set *R* includes the significant association rules which need to maintain, *k*, original table *T*, *QI*, the grid cell size.

Output: anonymous version table *T'*

```
1.   Create a grid and anonymize all location values into this grid.
2.   Construct a set S(satisfied groups), a set US(unsatisfied groups
3.   Sort the set US by above criteria.
4.   Calculate the values as in section 3.2 for each rule in R
5.   a set cannotProcess=∅, it contains groups that can not be
     transformed into a satisfied one.
     While (US is not empty){
6.      Select proUS from US by the priority degree
7.      US = US \ proUS
        While (proUS is still an unsatisfied group) {
8.         Run find_best_can_group() function to find a best change to
           transform proUS. A candidate group can and a set of tuples
           W containing tuples, which can be anonymized without
           affecting the goals, will be returned by this function.
9.         Exclude can from US or S
10.        if (can == null){cannotProcess = cannotProcess U proUS
11.           Give back all tuples, which are anonymized during the
              transformation of the current unsatisfied group, to
              their original groups.
12.           Unmark all examined groups in S and US
13.           break;
           } Else {
14.           Perform the change.
15.           Update support and confidence values of each rule in R
16.           Mark can as be examined
17.           if(can is satisfied group)   S = S U can
18.           Else US = US U can
19.           S = S U proUS
20.           Unmark all examined groups in S and US}}}
     if (cannotProcess is not empty){
21.     final_process()}
```

During the transformation of an unsatisfied group *proUS*, the algorithm will try to find changes which will apply to this unsatisfied group to transform this group into a satisfied one. Each change will have its cost which reflects the effect of this change on the goals. The cost for each change will be calculated in the *find_best_can_group()* function. From the costs of these changes, this function will also find the best changes for current unsatisfied group. A candidate group *can* and a set of tuples *W* containing tuples, which can be anonymized without affecting the goals, will be returned by this function. The set *W* will contain tuples from *can* if we have the change *can->proUS*. Otherwise, *W* will contain tuples from *proUS*. After receiving results from the *find_best_can_group()* function, the algorithm will perform the change, which is in accord with the results, for current unsatisfied group. After performing each change, if the unsatisfied group is not still satisfied, the algorithm will try to find additional changes to transform this unsatisfied group into the satisfied one. If the algorithm can not find any additional changes to transform the group without affecting the goals, this group will be moved to the set *cannotProcess*. The algorithm will try to solve this set at the final step. Clearly, the most important function is *find_best_can_group()*, which will try to find the best changes to transform current unsatisfied group into the satisfied one. As discussed above, the algorithm will try to maintain the significant association rules. Moreover, the algorithm will not generate additional significant

association rules, which their support values are greater than *t_s* and their confident value are also greater than *t_c*, during the running of it.

Name: *find_best_can_group()*
Input: unsatisfied group *proUS*, threshold *t_s*, threshold *t_c*
Output: a group *can* and a set *W* contains tuples that can be moved, the direction of the change (*proUS->can* or *can->proUS*).

```
1.   A group can = null
     For each temp from US U S (exclude proUS and examined groups) {
2.      Calculate the cost for changes: proUS-> temp and temp-> proUS
3.      Generate set W}
     If (exist the changes that do not violate the goals) {
4.      Choose a best change so that: (i) when performing it, the
        goals are not violated and (ii) it has the lowest cost. The
        change will include a group temp, a set W and a direction
        which determines proUS->temp or temp->proUS
5.      Assign can = temp. }
     Return can and W
```

This function will calculate cost for each change at the first step. Intuitively, we will choose the change that has the lowest cost. The cost calculated will be based on the following criteria: (i) The number of significant association rules which will be insignificant after performing the change; (ii) The number of significant association which will be generated after performing this change; (iii) The danger degree of significant rules after performing the change: for example, a significant rule has *support=0.7* and *confidence=0.6*. Assume that after performing the change number 1, this rule will have *support=0.64* and *confidence=0.53* and after performing the change number 2, the corresponding values will be *0.67* and *0.59*. The change number 2 will be better because it make the rule less dangerous; (iv) The number of tuples in the set *W*: the algorithm prefers set *W* which has greater number of its tuples because the more the number of tuples in the set *W*, the more satisfied an unsatisfied group.

After anonymization, there are some unsatisfied groups which the algorithm can not find the changes to transform these unsatisfied groups into satisfied ones. These groups will be added to the set *cannotProcess*. We also notice that before an unsatisfied group will be added to the set *cannotProcess*, all tuples, which are anonymized during the processing of this unsatisfied group, will be back to their original groups. It means that all groups will return the statuses which they had before transforming current unsatisfied group. In the case the set *cannotProcess* is not empty, the algorithm will run some additional steps to transform groups in this set into satisfied ones, these addition steps are in the *final_process()* function: At the first step, the algorithm will try to transform unsatisfied groups, which are in the set *cannotProcess*, into the better groups that are more satisfied than the original group. It also means that the number of tuples in each better group will be closer to *k* or 0. To do this step, the algorithm will choose the best changes, which will not affect the goals when performming them, to transform the unsatisfied group into a better one. The function *find_best_can_group()* can be used to find these best changes in this step. At the second step, the algorithm will try to transform these better unsatisfied groups into the satisfied ones. The algorithm will find changes that have the least effect on the goals. After that, it will perform these changes to transform the better unsatisfied groups into the satisfied ones. Different from the previous steps, the goals will be violated if these changes are performed. It means that some significant association rules may be no longer significant and/or new significant association rules may be generated after these changes are performed. This is “cost” which we must

pay to guarantee k-anonymity for the database because with these unsatisfied groups, the algorithm can not find any changes to transform them without effect on the goals.

5 Evaluations

In this section, we show the evaluation we conducted in order to evaluate the effectiveness of our algorithms. We will verify the proposed algorithm with three other algorithms: M3AR [2], KACA [7], OKA [6] in both criteria: the percentage of lost significant association rules and the percentage of new significant association rules that are generated during the run of algorithms. Intuitively, the smaller two values, the more effective the algorithm. We call them as *p_s* and *p_n*:

$$p\_s = \frac{l\_r}{t\_r} \tag{4}$$

$$p\_n = \frac{n\_r}{t\_r} \tag{5}$$

Where *l_r* is the number of significant association rules that are lost during the run of the algorithm, *n_r* is the number of significant association rules that are generated during the run of the algorithm and *t_r* is the total of significant association rules.

The real database, which is used for the evaluation, will be extracted from GeoLife project [4], which is collected in (Microsoft Research Asia) GeoLife project by 165 users in a period of over two years (from April 2007 to August 2009) and Adult database from the UC Irvine Machine Learning Repository [5]. This database will include 34827 records. The *QI* will include status, age, sex and location attribute. The grid cell size, which is used to anonymize the location attributes, is 500m*500m. For each value of *k*, we will execute each algorithm in five times; the achieved result is the average of five tests. The following figures show the result of the evaluation.

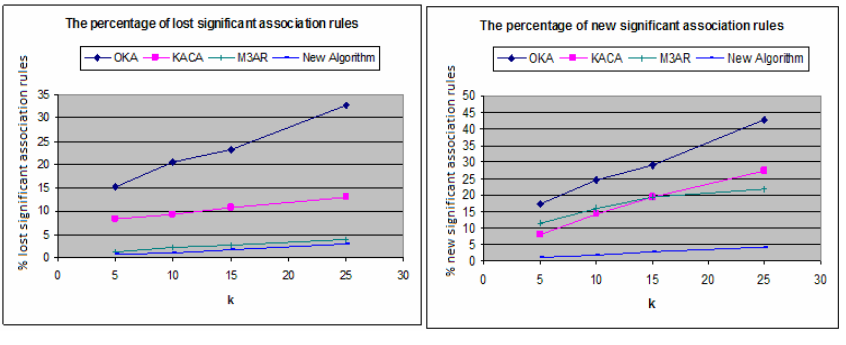


Fig. 1. The evaluation results

These results show that with our proposed algorithm, the percentage of significant association rules, which are lost during the run of the algorithm, is minimal. Similarly, the percentage of new significant association rules, which is generated during the processing, is also minimal. It also means that our algorithm will generate an effective k-anonymous version of the database. The reason of these results is that our algorithm

tries to transform the unsatisfied groups with the changes that will cause least effect on the goals. Therefore, the result of data mining process may be more effective.

6 Conclusion and Future Works

In this paper, we proposed an algorithm that anonymizes the location database to an effective k-anonymous version. The algorithm solves some problems in the M3AR algorithm that was proposed before to guarantee k-anonymity for general databases. With the algorithm, the number of significant association rules, that are lost during the anonymization, is reduced. Moreover, the number of significant association rules, which are generated during the anonymization, is also reduced. Thus, the results generated by the data mining process, which input data is the k-anonymous version of the database, are more effective and more valuable. We also applied the grid based solution to reduce the number of significant association rules and also reduce the number of unsatisfied groups. Thus, the algorithm is more effective.

In the future, we will focus on investigating additional solutions to improve the performance of the algorithm. On the other side, we should improve the criteria which are applied to assign a priority degree for each unsatisfied group so that the algorithm can return a more effective k-anonymous version of the database. Moreover, the location of the user is usually accompanied with a time value. Therefore, the algorithm should also consider the time value when anonymizing the database.

References

1. Truong, Q.C., Truong, T.A., Dang, T.K.: Memorizing Algorithm: Protecting User Privacy using Historical Information of Location-Based Services. IJMCMC 2(4), 65–86 (2010)
2. Dang, T.K., Küng, J., Huynh, V.Q.P.: Protecting User Privacy while Discovering and Maintaining Association Rules. In: 4th IFIP International Conference on New Technologies, Mobility and Security, France, pp. 109–118 (2011)
3. Sweeney, L.: k-Anonymity: A Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 557–570 (2002)
4. Zheng, Y., Li, Q., Chen, Y., Xie, X.: Understanding Mobility Based on GPS Data. In: ACM Conference on Ubiquitous Computing, Korea, pp. 312–321 (2008)
5. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases (1998), `http://www.ics.uci.edu/mlearn/MLRepository.html`
6. Jun, L.L., Meng, C.W.: An Efficient Clustering Method for k-anonymization. In: Int. Workshop on Privacy and Anonymity in Information Society, France, pp. 46–50 (2008)
7. Li, J., Wong, R.C.W., Fu, A.W.C., Pei, J.: Achieving k-Anonymity by Clustering in Attribute Hierarchical Structures. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 405–416. Springer, Heidelberg (2006)
8. To, Q.C., Dang, T.K., Küng, J.: B^{ob}-Tree: An Efficient B+-Tree Based Index Structure for Geographic-aware Obfuscation. In: Nguyen, N.T., Kim, C.-G., Janiak, A. (eds.) ACIIDS 2011, Part I. LNCS (LNAI), vol. 6591, pp. 109–118. Springer, Heidelberg (2011)
9. Van Quoc, P.H., Dang, T.K.: eM^2: An Efficient Member Migration Algorithm for Ensuring k-Anonymity and Mitigating Information Loss. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 26–40. Springer, Heidelberg (2010)

Smeagol: A "Specific-to-General" Semantic Web Query Interface Paradigm for Novices

Aaron Clemmer and Stephen Davies

University of Mary Washington, Fredericksburg, VA 22401, USA
{aclemmer,stephen}@umw.edu

Abstract. Most Semantic Web query interfaces let the user give an abstract specification of the desired results (perhaps using facets, or a natural language query.) We introduce the Smeagol visual query interface which, by contrast, guides the user from a specific example to a general result set. Users begin the query process with navigation and exploration activities, building a concrete subgraph of interest from the larger data set. They then generalize this subgraph to find other subgraphs similar in some way to the one identified. Among other advantages, this approach also lends itself quite naturally to querying on instance-based data; i.e., triples in which the predicate is not part of a defined ontology. We provide an analysis of this specific-to-general approach, contrasting it with existing systems. We also present the results of a usability experiment comparing novices' use of Smeagol with that of a standard Linked Data browser.

Keywords: Semantic Web, linked data, user interface, query building, pivot operation, graph visualization.

1 Introduction

In the last few years, Semantic Web researchers have begun to produce interfaces that enable novices to pose queries without the use of a formal query language. Some of these applications accept natural language queries; others let the user directly manipulate a graphical representation.

One thing these diverse systems share is an interface paradigm that progresses from the general to the specific. Users give an abstract specification of the desired results, whether by using facets, natural language description, or some other means. Although the mechanism used to express the query varies widely, the user's task is ultimately still to express some variation of the general formula "find resources that satisfy these criteria."

A subtle problem is that this approach often does not mimic the user's thought process. Sometimes the user may begin with an abstract question in mind, but often he does not. Instead of knowing at the outset that he wants to ask "Who are all the famous athletes who dated celebrities?" a user may be browsing a David Beckham page, discover that Beckham dated Victoria Adams, and think, "Interesting! I wonder what other athletes were similar?" Only after asking that question, in those circumstances, will he discover that Derek Jeter dated Mariah Carey. In other words, a user often does not even realize that he *has* a question until an intriguing concrete example is found.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 288–302, 2011.

We believe there may be an advantage to an interface that explicitly enables this process. With such a tool, users could roam and explore a Semantic Web data set without regard to any possible future query. As they browse, they mark out features of interest along the way, building a subgraph which illuminates a small subset of resources and relationships. Then, they generalize from this example in whatever way(s) they choose in order to see analogous resources.

Even when users *do* begin with a question, they may benefit from being able to express that question by means of an example. Rather than having to begin abstractly with a list of types and predicates, users can find a concrete example of what they are looking for, and construct the query "in place." This seems less error-prone, since the user is directly working with the very predicates and graph structure for which they want results, rather than having to describe the desired instances in general terms. This approach also naturally supports querying on *any* predicate, not merely those defined by an ontology. It is possible, of course (see, e.g., VisiNav[6]) to design a faceted interface that exposes non-ontology-based predicates, but it seems that the context in which one first discovered the existence of a predicate is a quite natural place from which to select that predicate and find other examples.

We define the term "general-to-specific" to refer to a query interface (like Humboldt[12] or gFacet[7]) that allows the user to specify abstract criteria for a result set. By contrast, we define "specific-to-general" to refer to an interface that explicitly supports starting with an example and generalizing it to find other similar examples. Put another way, a "general-to-specific" interface is based on reduction: adding criteria to the query progressively narrows down the results from the set of all resources to the desired set of answers. A "specific-to-general" interface, on the other hand, is based on expansion: aspects of a concrete example are progressively generalized to find other results that match a pattern.

To continue the above example, a general-to-specific interface would allow a user to find the class of Footballers (or Professional Athletes) in a data set, then choose from predicates like "dated," "marriedTo," or "inARelationshipWith." The user could form a query based on such predicates, and add additional constraints, such as that the object of the triple must be of a certain type (Celebrity.) This would allow the user to find both the Beckham-Adams and Jeter-Carey query results, but only by beginning with (and knowing about) the abstract types and predicates, and anticipating that there would be some result(s) that satisfied them. By contrast, a specific-to-general interface would allow the user to browse the data set, and upon reaching a subgraph reflecting the Beckham-Adams relationship, immediately request to generalize that relationship to others that followed the same pattern. No top-down selection of classes, predicates, or desired subgraph patterns is required, since these are directly under the user's nose at the time the query is generated. There is also no need for the user to guess what the "correct" predicates or classes are, since the example immediately in front of the user already contains them.

This paper introduces Smeagol, a query interface that directly supports the specific-to-general paradigm. In J.R.R. Tolkien's mythology, Smeagol was a

creature who found something of great value without deliberately setting out to search for it. Similarly, our application allows the user to begin the query process with navigation and exploration activities, building a concrete subgraph of interest from the larger data set. They then generalize this subgraph to find other subgraphs similar in some aspect(s) to the one identified. Our theory is that ordinarily, a novice user's "queries" arise in exactly this way: not as a quest for general results satisfying some abstract criteria, but as a search for other items analogous to a concrete particular.

Note that our work is about identifying, designing, and empirically evaluating a new user interface paradigm. There are many other open questions about querying the Web of Linked Data which we do not address. These include: performance and scalability of complex queries; evaluating queries that span multiple data sets and hence require data integration to satisfy; and discovering relevant data sets. These are all important open problems, and we are aware that others are working on them. They are, however, outside the scope of our research. Our focus is on enabling novice users to effectively pose complex queries against a Semantic Web data set, a challenging task given the complex nature of graph-based data and the difficulty many humans have visualizing and articulating patterns in it.

2 Related Work

There are a variety of user interfaces that enable novices to query the Semantic Web. These differ from search interfaces such as Sindice[16], Swoogle[5], and Falcons[2], whose purpose is to find resources matching a keyword or property, in that they enable the user to answer complex questions expressed as a graph pattern, such as "Who are all of the authors of books published in Germany in the year 1974?"

Within the domain of query interfaces, Natural Language Interface (NLI) systems such as FREyA[4], PowerAqua[13], PANTO[17], and SerFR[1] focus on helping the user find an answer to an *a priori* question, as opposed to supporting an iterative process of domain discovery and query building.

Visual query builder (VQB) systems, such as iSPARQL[15], Semantic Crystal[11], NIGHTLIGHT[14], and SPARQLinG[8], manifest the query as a graph (as Smeagol does), but the top-down emphasis on constructing formal graph patterns from an ontology expects significant knowledge of both the problem domain and SPARQL. Neither NLIs nor top-down VQBs are oriented towards scenarios where a formal ontology does not exist.

Faceted interfaces, such as Humboldt[12], Parallax[9], gFacet[7], and VisiNav[6], are more oriented towards novices than VQB systems. Like Smeagol, they enable novice users to both explore the problem domain and also iteratively build a query, particularly when they do not begin with a clear *a priori* question. However, faceted interfaces are intrinsically general-to-specific, because the user starts with a generalized set of resources (perhaps all resources of a given type) that is reduced by selecting facets which filter the set. That is, facets are abstract criteria that narrow the result set.

In addition to letting a user filter results on simple properties, Humboldt and Parallax allow the user to "pivot" to properties of related objects (of different types.) For example, a user querying for automobile manufacturers can pivot to the set of automobiles manufactured by Toyota. This implementation of the pivot concept is powerful, but has three important limitations. (1) Only a portion of the query specification is visible to the user at any given time, due to the selected facets only being displayed on the respective pages they were selected from. This was noted by [7], and we believe that grouping the constraints of the entire query together would place less cognitive burden on the user, since they could then see a unified presentation of the query. (VisiNav[6] is an example of a faceted interface which addresses this limitation.) (2) Not all types of queries (which we term "query topologies"; see Section 5) are supported by these interfaces because they utilize a linear history model (i.e. the past sequence of user pivots.) This limitation was noted by [12] for Humboldt but applies to Parallax as well. An example of a non-linear query that cannot be posed by these interfaces is "What musicians contributed to a 2010 album, and also wrote a book of poetry?" which requires a branching sequence of pivots to specify all relations and facets. (3) When the user wants to view results across multiple pivots, he must visit each relevant page in the history to assemble the results. That is, the user can only view one column of the query results at a time. Smeagol addresses all three of the above concerns by (1) manifesting the entire query in a single display, (2) supporting arbitrary branching topologies, and (3) presenting all query results as a set of tuples in a unified display.

gFacet[7] is a facet graph interface, and so uses the general-to-specific paradigm as all faceted interfaces do. It is similar to Smeagol in that the user's query is represented visually as a graph, but there are several key differences. First, in gFacet the facets are derived solely from ontology[1], so it is not possible to express queries involving arbitrary predicates. Second, it does not provide a facility for viewing all statements made about a given resource, limiting the ability of the user to explore the domain to discover what kinds of queries can be posed. Third, like Humboldt and Parallax, the user cannot see a unified, assembled result set across multiple pivots.[2]

The MashQL interface[10] is essentially a dynamic, hierarchical, form-based query builder. It implements pivots and supports both ontology and arbitrary

[1] For DBpedia, the facets are `skos:subject` objects of type `skos:Concept`, paired with the predicates relating them back to the subject of the triple whose resources have `skos:subject` as the current facet.

[2] For example, suppose the user has a query graph with three facets: song titles in the category `Songs_written_by_John_Lennon`, producers of those songs who are `LivingPeople`, and record labels of those songs that are `RockRecordLabels`. If the user wanted to know all songs produced by Yoko Ono and their respective record labels, he would first have to select Ono from the list of `LivingPeople`. He would then see the songs produced by Ono and the record labels of those songs, but the two lists would not be correlated with each other. In order to correlate the two, the user would need to click on each record label resource in turn to determine which songs correspond to it.

properties, all composed in a tree-based view. The user progressively specifies graph patterns using dynamically constructed dropdowns whose contents reflect the current context (e.g. if a subject dropdown is set to a particular resource, the predicate dropdown will contain all properties used by that subject). Pivots are expressed through the hierarchical representation of the graph patterns, and the user can then specify which elements will be returned in the results. MashQL largely utilizes the general-to-specific paradigm, as the user specifies a general pattern using subject classes as well as predicates, which may return many results. He may then reduce the results to a specific answer by either adding further relations, or by specifying that a property's subject or object has a particular value. Note that the interface also does support the specific-to-general paradigm to a degree, as concrete concepts can be chosen from dropdowns rather than from an ontology. However, the interface's form-based query building encourages users to think from a top-down perspective, and it does not provide a straightforward way to view all statements made about a particular resource, limiting the exploration required to locate conrete resources.

In summary, these query interfaces comprise numerous powerful features that enable users to pose queries in intuitive ways. However, none of them support the specific-to-general paradigm in the way that Smeagol does.

3 The Smeagol User Interface Paradigm

Smeagol supports a threefold procedure for building queries. Each step is intended to lead naturally to the next.

1. **Exploration.** Users begin by exploring the data graph, traversing from resource to resource and seeing the statements made about each one. This is similar to most Linked Data browsers, but different from most query interfaces. It enables the user to begin with a familiar navigation task, traversing from concrete resource to concrete resource. The user can inspect the triples involving each resource of interest. During this process of inspection and traversal, the user does not face a burdensome cost in terms of backtracking or reorientation when pursuing casual exploration or encountering deadends.

2. **Subgraph building.** As the user begins to identify an area of interest, he can select particular triples and add them to an "example subgraph." This is a connected subset of the overall Web of Data that reflects a user's current focus. It consists of a handful of specific resources and the relationships between them. Building this subgraph serves two purposes: (1) it lets the user select and highlight only the relevant subset of information out of the overwhelming amount of data in the overall graph, and (2) it forms the basis for a future query.

We believe this step is particularly important as users transition from today's free-text-based Web to the Semantic Web. In the free-text Web, the content of a single page contains so much context that visualizing its neighbors is not as crucial. But when browsing Linked Data, where each node represents a succinct nugget of information, a user can quickly become disoriented if he cannot visualize the contextually pertinent relationships around it.

3. **Subgraph generalizing.** Finally, the interface assists the user in generalizing his specific example into a query, so that he may view the answer(s) to a question he wants to pose. From the specific example, the user can express that certain concrete subjects and objects are actually the resources that he wants to generalize from; that is, they are the variables of the query. The properties and remaining resources are considered to be the constraints.

Though presented here as a sequence, Smeagol users can naturally move back and forth between these activities as they explore the data graph. Adding more triples to the subgraph moves from activity 3 back to activity 2, and navigating to a resource moves back to activity 1. In this way, queries can be modified, expanded, and refined in an interactive process.

3.1 User Interface

In order to begin exploring a graph of Semantic Web data, the user must specify a URI to use as his starting point. Smeagol provides a simple search interface that utilizes DBpedia's URI Lookup web service[3]. The user types one or more keywords into a search box, which when submitted gives a list of suggested URIs for consideration. The user must select the URI from which he wishes to begin exploring; this URI becomes the first resource in his subgraph. Once the user chooses, he is taken to the main Smeagol interface.

The Smeagol interface (Fig. 1) is divided into three sections. The inspector pane (left) displays all triples involving the user's current resource of interest in an "infinite-scrolling" list. When the user decides to add a particular triple to his subgraph of interest, he can identify it as such by selecting it in the list. Conversely, if the user changes his mind, he can remove it from his subgraph by unselecting it. Smeagol currently has no mechanism for intelligently limiting the number of triples in the inspector; this is a difficult problem, and is a topic for future work.

The query visualizer pane (top-right) displays the user's current subgraph. Selections and unselections made in the inspector pane immediately result in an animated update of the visualization. The subgraph is depicted using a radial layout algorithm. The advantage is one of locality: the resource in the center of the visualization is the one currently most relevant to the user; it is also the resource shown in the inspector pane. The distance from the center resource to another resource reflects the degree of separation between them; a distant resource is usually less important to the user than a direct relation. The user may choose to shift his focus and inspect a more distant resource by clicking on its name in the query visualizer; this moves that resource to the center. The radial layout may be panned by clicking and dragging, which provides access to resources too distant to be in immediate view. A resource in the query visualizer pane can be removed from the subgraph, or "wildcarded" (generalized) via a pop-up menu. Choosing to remove the resource from the query will remove it from the user's subgraph, and also prune the subgraph at that point. This behavior

[3] http://lookup.dbpedia.org/

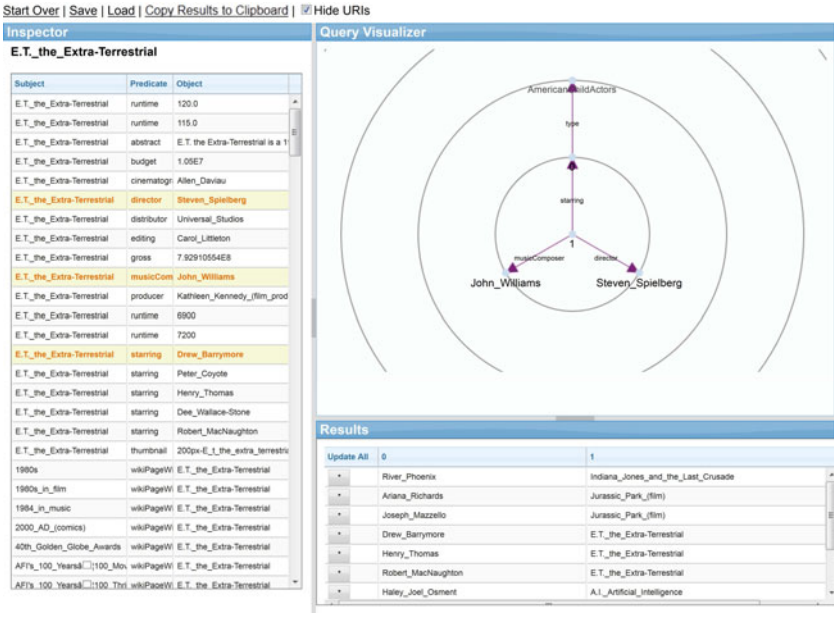


Fig. 1. The Smeagol User Interface

simplifies the process of trimming unnecessary paths of exploration from the subgraph.

Fundamental to the subgraph building procedure, the user can generalize his subgraph at any time by choosing to wildcard a resource; any number of resources may be wildcarded. (When a wildcarded resource is displayed in the inspector, its last associated concrete resource is shown.) The act of wildcarding a resource results in a query being executed.

The results pane (bottom-right) displays a table of tuples corresponding to the query results, which refreshes whenever the wildcarded state of a resource in the query visualizer changes. This state change causes the subgraph to be translated into a SPARQL query, where each variable in the query corresponds to both a wildcarded resource in the query visualizer and a column in the results pane.

There are two operations the user can perform from the results pane. Clicking on a cell in the table replaces the corresponding wildcarded resource in the subgraph with the cell's value. This results in a new query being run and the results pane being refreshed. Clicking on a row's "Update All" button replaces all wildcarded resources in the subgraph with the respective concrete resources in that row of the results. This effectively replaces the entire query with a chosen concrete example. The benefit of these operations is to aid in the explorative process: if the user discovers an interesting resource in the results, he not only can update the query to restrict on that resource, but can also inspect it and modify the subgraph based on what is seen.

3.2 Example

To provide an example of the process of exploration, subgraph building, and subgraph generalization with Smeagol, consider the following: Suppose the user searches for the 1982 film *E.T.: The Extra-Terrestrial*, and after selecting it, he is taken to the main Smeagol interface. At this point, the center resource in the query visualizer pane would be E.T. (Fig. 2, A), and the inspector pane would show all statements about the film. Browsing through the inspector, the user notices that the film was directed by Steven Spielberg, the music was by John Williams, and it starred Drew Barrymore. Each of these statements interests the user, so he clicks on them in turn in the inspector pane to add them to the query visualizer pane (Fig. 2, B).

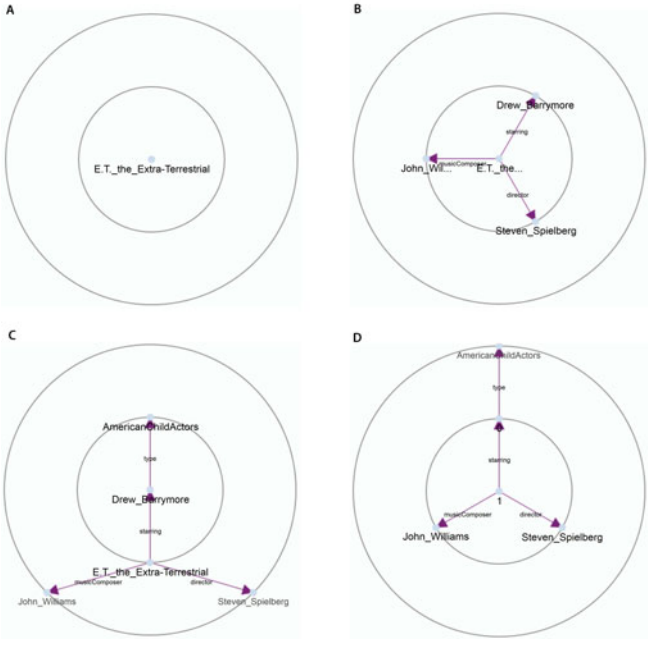


Fig. 2. The query building process

Deciding to explore Drew Barrymore, the user clicks on that resource in the query visualizer pane, which causes the graph to move the resource to the center and the inspector pane to load information about the actress. Browsing through the inspector, the user sees that she is of type `AmericanChildActors`. Recalling that Barrymore was quite young in E.T., a question occurs to the user: did Stephen Spielberg direct any other films which starred an American who at one time was a child actor? Moreover, did Spielberg and Williams collaborate on any such films? To answer this query, the user begins by adding the `AmericanChildActors` resource to the query visualizer (Fig. 2, C). He then

chooses to wildcard both the E.T. and Drew Barrymore resources (Fig. 2, D), as he wishes to see more films and actors meeting the above criteria. After doing so, he is presented with the results as seen in Fig. 1.

4 Architecture

Smeagol is a Rich Internet Application (RIA) that communicates via REST web services to a Java server application. The server application provides proxying to SPARQL endpoints, query result caching, and persistence of users' query graphs.

Smeagol has been tested against DBpedia's SPARQL endpoint, but is not architecturally limited to it. The SPARQL query used by the inspector makes no assumptions about ontology or the presence of any DBpedia-specific resources:

```
SELECT DISTINCT ?subject ?predicate ?object WHERE {
  {<resource> ?predicate ?object}
  UNION
  {?subject ?predicate <resource>}
  FILTER (lang(?object) = "en" || lang(?object) = "")
} ORDER BY ?predicate ?subject
```

The SPARQL queries generated by the query builder from the user's example subgraph are similarly independent of DBpedia. Note that Smeagol could be adapted to data sources other than SPARQL endpoints if those sources could be accessed in a programmatic way, since the user does not deal explicitly with SPARQL.

The query builder currently supports a subset of SPARQL syntax: the user may specify triple patterns and bind variables. More advanced syntax was not necessary for the initial validation of the specific-to-general query paradigm.

Performance of the application is determined by the responsiveness of the SPARQL endpoint and the complexity of the queries the user chooses to construct within the query visualizer. The inspector queries are simple, an advantage afforded by the specific-to-general approach is a reduction in ontology-related queries needed to drive user exploration and query formulation, compared to general-to-specific interfaces. Finally, paging is used to manage large result sets returned by inspector and user queries.

5 Query Topologies

In order to identify which kinds of queries Smeagol confers an advantage for, we define the notion of a *query topology*. A query topology describes the general structure of a subgraph in terms of the nodes and the relationships between them, including which of the nodes are wildcarded. It essentially characterizes a certain class of triple patterns.

We depict query topologies visually as a graph of nodes, where "R" indicates a particular concrete resource, and "*" indicates a wildcard. (See Fig. 3.) This is similar to a SPARQL triple pattern – with R's as URIs and *'s as variables – except that we are generalizing from any particular pattern to a category of all structurally similar patterns. Here is one query that conforms to this topology:

```
SELECT * WHERE {
    :Kenneth_Branagh  :starringIn  ?film
    ?film  :writer  :J._K._Rowling
}
```

This query conforms to the topology since it contains one variable present in two triples, each of which also contains one concrete resource. Note that a query topology diagram is an *un*directed graph, since from a complexity standpoint it turns out to be immaterial whether a given node in a triple is a subject or an object. (True, the Semantic Web is a directed graph, since each triple has a subject and object, but the queries "Spielberg directed ?x" and "?x directedBy Spielberg" are equally difficult to pose and to evaluate, which is all we are concerned with.)

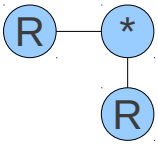


Fig. 3. A query topology

We characterize a query topology by a three-numbered designation *(n-w-l)*, where n is the total number of nodes in the graph, w is the total number of wildcards, and l is the length of the longest path of consecutively wildcarded nodes, excluding branches. We choose these three measures, omitting other features of the topology (for example, whether two R nodes are attached to the same * node, or to different ones) because they represent likely elements of user difficulty. The number of nodes, the number of generalized nodes, and the "density" with which the generalized nodes are glued together all represent different aspects of a query's complexity. We hypothesize that the third of these three quantities will be particularly significant, since it essentially captures the number of pivots necessary to execute the query. The topology in Fig. 3 is of class (3-1-1).

Further examples of query topologies, along with their topological designations, sample SPARQL queries, and sample English questions, are in Fig. 4.

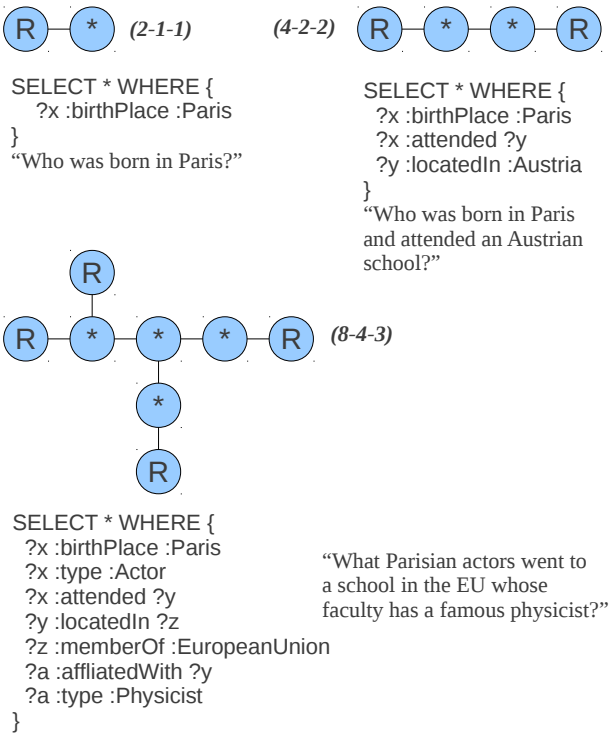


Fig. 4. Some query topologies, with designations, sample SPARQL queries, and sample English questions

6 Usability Experiment

From a user's perspective, Smeagol supports the same kind of open-ended exploration that a Semantic Web browser does, but adds the ability to naturally transition to a query task. Hence, to test whether this added ability yields any benefits, we conducted a usability experiment to compare novices' performance in using Smeagol with that of a Linked Data browser. Our goals were to determine whether the Smeagol subgraph-building and wildcarding paradigm was operable by novice users, and to identify which query topologies it gave an advantage for.

Our subject pool consisted of 43 undergraduate college students, ranging from 18 to 22 years of age and containing roughly an even split between genders. All students were enrolled at the University of Mary Washington during the Fall 2010 semester and were of many diverse majors.

Participants took the one-hour experiment using the Firefox Internet browser on a Windows workstation. They first received a ten-minute explanation of Semantic Web concepts, and a demonstration of the particular navigation tool they were to use (depending on the group; see below.) They then received a packet

of materials containing twelve timed query tasks, and were directed to a URL to begin. Both navigation tools used the DBpedia SPARQL endpoint as the sole data source.

Our control group used the Pubby Linked Data frontend [3], which provides a simple HTML interface to the DBpedia SPARQL endpoint. Pubby's interface is similar to many Linked Data browsers in that it manifests each resource as a page, and shows all triples that have that resource as either a subject or object. This is nearly identical to Smeagol's inspector pane, except that full URIs are shown for each resource rather than just its label. Our experimental group used the Smeagol application as described in Section 3.1.

Each item in the packet contained an English language question which the participant was instructed to find the answer to. To begin, the participant was directed to a certain resource that was one of many "answers" to the question. For the control group, this was simply the Pubby page for the starting resource. For the experimental group, the participants began by loading a stored Smeagol subgraph with that node present. For example, item D asked "What famous authors went to Harvard Law School?" and users began the item from the Barack Obama Pubby page (control) or from a subgraph containing the Barack Obama node (experimental). In this way, we hoped to simulate the process described at the beginning of this paper: a user beginning with a concrete example and wanting to generalize it to other examples.

The items were grouped into two sections. The first section contained "scripted" items which explicitly provided the user with the required subgraph. In item D, for example, the control group was told which predicates and resources were relevant (`:almaMater`, `:Harvard_Law_School`, `:occupation`, and `:Author`) and explicitly directed how to find them and what to click on. The experimental group's starting subgraph contained all of these nodes. The participant's task, then, was not to determine how to build the subgraph, but simply to generalize the subgraph: either by navigating and filtering (control) or by wildcarding nodes and noting the results (experimental.)

The second section of the packet contained *non*-scripted items. For example, item J asked "What U.S. Democrats who participated in World War II battles went to a college in the Ivy League athletic conference?" and simply started each participant on the John F. Kennedy page (control) or with a subgraph containing only the John F. Kennedy node (experimental.)

Both sections featured items of various topological classes. Item D (above), for instance, was of class (3-1-1), and item J was (5-2-2). Through this variety we aimed to isolate the different cases and quantify how well Smeagol improved user performance in various scenarios.

7 Results

The results for each item in our hour-long experiment are presented in Table 1. The "control" column indicates how many users in the Pubby group got each item correct in the time frame allotted (generally 4-6 minutes, depending on

Table 1. Results for each experimental item, including tallies and percentages for each group (control and experimental) who correctly completed the item. Bold p-values indicate statistically significant results (to $\alpha = .05$ by two-tailed Fisher's exact test.)

Item	Topology class	Scripted?	control	%	exp	%	p-value
A	(2-1-1)	yes	17/23	73.9%	19/20	95.0%	.100
B	(2-1-1)	yes	19/23	82.6%	20/20	100.0%	.111
C	(2-1-1)	yes	19/23	82.6%	18/20	90.0%	.669
D	(3-1-1)	yes	0/23	0.0%	17/20	85.0%	**<.0001**
E	(4-2-2)	yes	0/23	0.0%	7/20	35.0%	**.002**
F	(2-1-1)	no	17/23	73.9%	18/20	90.0%	.250
G	(2-1-1)	no	20/23	87.0%	20/20	100.0%	.236
H	(3-1-1)	no	0/23	0.0%	15/20	75.0%	**<.0001**
I	(3-1-1)	no	4/23	17.4%	12/20	60.0%	**.005**
J	(5-2-2)	no	0/23	0.0%	13/20	65.0%	**<.0001**

item complexity) and the "exp" column shows the same for the Smeagol users. An answer was deemed to be "correct" if its list of resources satisfactorily answered the question for some reasonable choice of predicates. In some cases, more than one reasonable choice existed (such as `:sports` and `:affiliation` for "Ivy League" schools in question J), and so correct answers sometimes varied from one another.

The first and most obvious finding is that the Smeagol group outperformed the Pubby group on every item. However, this was statistically significant (using Fisher's exact test rather than χ^2 due to small sample sizes) only for items in which the number of nodes is greater than two. A simple thought experiment reveals the likely reason for this. In a (2-1-1) topology – for instance, "Which films did Tom Hanks produce?" (item C) – a Pubby user can navigate to a single page (Tom Hanks) and examine the predicates to find a list of results. On the other hand, in a (3-1-1) topology – for instance, "What famous authors went to Harvard Law School" – a Pubby user faces a nearly hopeless navigation task. Starting from either the "Author" page or the "Harvard_Law_School" page, they can only identify a list of *possible* results. They must then manually navigate to each author (or Harvard alumnus) to determine whether the other criterion is satisfied. Smeagol, of course, makes such navigational legwork unnecessary through the use of wildcards.

We suspect, but did not verify in this experiment, that the problem for Pubby users would be exacerbated for queries involving a longer chain of wildcards (i.e., those with a larger value of l in the *(n-w-l)* designation.) This effectively multiplies the number of traversals exponentially. For instance, item E – "What Chicago Bears football players went to college in the Big East?" – requires a Pubby user to find players from the Chicago Bears page, but thereafter to face a combinatorial explosion. Each player cannot simply be checked for a property; rather, *all* schools that player was affiliated with must *each* be checked for a property, requiring a second set of multiple traversals. The total size of the problem varies with the average number of triples per predicate, of course, but

it quickly becomes unmanageable (to say nothing of time-consuming) even for a disciplined user. We plan to investigate this in future research.

8 Conclusions and Future Work

The specific-to-general query paradigm seems to be understandable by novices and beneficial to them. Equipping users with the ability to identify a subgraph and generalize it allows them to answer a much wider variety of questions than they could with only a standard navigational browser. This appears to be particularly true for queries that reach a certain threshold of complexity, where "complexity" involves both the number of nodes in a subgraph and the number and arrangement of wildcard nodes. Also, there is a benefit to users explicitly building and visualizing a concrete subgraph, so that they can better comprehend the immediate context of their inquiry.

Smeagol itself could be improved by enabling quantitative comparisons in queries, and by expanding the set of logical primitives to include unions and "ors." Additionally, we believe there may be benefit to organizing the predicates available for users to choose from by leveraging available ontology – this would complement the strengths Smeagol has in supporting queries built from arbitrary triples.

Having established a baseline against the standard Linked Data browser model, we plan in future work to compare novices' performance with Smeagol versus general-to-specific query interfaces like gFacet[7]. This should help us understand which use cases have the greatest benefit from a specific-to-general model; it is very possible that different sorts of user scenarios are better served by different approaches.

Smeagol is completely open source under the GPL license and source code is available at `http://bitbucket.org/aclemmer/smeagol`. A live demo of the application can be accessed at `http://rosemary.umw.edu/smeagol`.

References

1. Al-Muhammed, M.J., Embley, D.: Ontology-based constraint recognition for free-form service requests. In: Proceedings of the 23rd IEEE International Conference on Data Engineering, pp. 366–375 (2007)
2. Cheng, G., Ge, W., Qu, Y.: Falcons: searching and browsing entities on the semantic web. In: Proceedings of the 17th International Conference on World Wide Web at WWW 2008 (2008)
3. Cyganiak, R., Bizer, C.: Pubby – a linked data frontend for sparql endpoints (2007), `http://www4.wiwiss.fu-berlin.de/pubby/`
4. Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. The Semantic Web: Research and Applications, 106–120 (2010)
5. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, pp. 652–659 (2004)

6. Harth, A., Buitelaar, P.: Exploring Semantic Web Datasets with VisiNav. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554. Springer, Heidelberg (2009)
7. Heim, P., Ertl, T., Ziegler, J.: Facet graphs: Complex semantic querying made easy. The Semantic Web: Research and Applications, 288–302 (2010)
8. Hogenboom, F., Milea, V., Frasincar, F., Kaymak, U.: RDF-GL: a SPARQL-Based graphical query language for RDF. Emergent Web Intelligence: Advanced Information Retrieval, 87–116 (2010)
9. Huynh, D., Karger, D.: Parallax and companion: Set-based browsing for the data web (2009)
10. Jarrar, M., Dikaiakos, M.: A Data Mashup Language for the Data Web. In: Proc. of Linked Data on the Web (LDOW 2009) Workshop at WWW 2009 (2009)
11. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? In: Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, pp. 281–294 (2007)
12. Kobilarov, G., Dickinson, I.: Humboldt: Exploring linked data. In: Proc. of Linked Data on the Web (LDOW 2008) Workshop at WWW 2008 (2008)
13. Lopez, V., Motta, E., Uren, V.: Poweraqua: Fishing the semantic web. The Semantic Web: Research and Applications, 393–410 (2006)
14. Smart, P., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.: A visual approach to semantic query design using a web-based graphical query designer. Knowledge Engineering: Practice and Patterns, 275–291 (2008)
15. Software, O.: OpenLink iSPARQL (2010), http://demo.openlinksw.com/isparql/
16. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, pp. 552–565 (2007)
17. Wang, C., Xiong, M., Zhou, Q., Yu, Y.: Panto: A portable natural language interface to ontologies. The Semantic Web: Research and Applications, 473–487 (2007)
18. Zloof, M.M.: Query-by-example: a data base language. IBM Syst. J. 16(4), 324–343 (1977), http://portal.acm.org/citation.cfm?id=1662134

Browsing-Oriented Semantic Faceted Search*

Andreas Wagner, Günter Ladwig, and Thanh Tran

Institute AIFB, Karlsruhe Institute of Technology, Germany
{a.wagner,guenter.ladwig,ducthanh.tran}@kit.edu

Abstract. Faceted search enables users to browse and discover relevant items from a large collection such as the Web of data. Existing faceted search solutions assume a precise information need, and thus optimise relevance, interestingness, and costs of fulfilling an information need. In this paper, we propose a complementary solution. Instead of assuming a *search scenario* (i.e., a user has a precise information need), our solution targets a *browsing scenario* (i.e., a user has a fuzzy need). We aim to support users in exploring an unknown collection of items, thereby allowing them to discover new or unfamiliar items of interest. Our approach comprises mechanisms for *grouping facets and facet values* and facet *ranking*. Via a task-based evaluation, we demonstrate that the proposed solution enables more effective browsing compared to the state-of-the-art, given fuzzy information needs.

1 Introduction

Recently, large amounts of structured data have been made publicly available on the Web (e.g., RDFa[1] or Linked Data[2]), allowing complex information needs to be addressed. For instance, consider the following example: Susan is a novice computer science student. She is eager to learn more about this vast research field and wishes to find "information about work of prestigious computer scientists". With traditional Web search, Susan searches via keywords and browses via hyperlinks to fulfil her information need. Observe the two paradigms in Susan's example: (1) *search* as a mean for goal-oriented retrieval of information (e.g., via keyword-based lookups) and (2) *browsing* as a mean for iterative exploration of a collection of items (e.g., via hyperlinks) [18,17].

Searching Web data using structured query languages (e.g., SPARQL[3]) helps to address complex information needs (e.g., Susan's need). However, in order for such a goal-oriented search to be effective, users have to be familiar with the query language. Further, users have to know the item of interest and the underlying domain. Thus, the search paradigm allows for *precise* information needs only. However, real-world information needs are often *fuzzy*. There are

* Supported by the German Federal Ministry of Education and Research in the *CollabCloud* (grant 01IS0937A-E) and *iGreen* (grant 01IA08005K) project.

[1] http://www.w3.org/TR/xhtml-rdfa-primer/

[2] http://www.w3.org/DesignIssues/LinkedData.html

[3] http://www.w3.org/TR/rdf-sparql-query/

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 303–319, 2011.

two dimensions of fuzziness: (1) Users have *vague knowledge about the domain*. For instance, Susan cannot precisely describe the term "prestigious", whereas a domain expert (having precise knowledge) may look for researchers that won a Turing Award. (2) Users only *vaguely know the item of interest*. For instance, "work" in our example may refer to publications or projects.

The browsing paradigm is more suitable for dealing with fuzzy needs. It does not assume users to have full knowledge regarding domain or item of interest. Instead, browsing allows users to explore a collection of items iteratively [18,17]. For instance, Susan may start with a simple lookup search (e.g., a keyword query "computer scientist") to obtain some starting points and then browses the remaining collection to find "prestigious" scientists.

Faceted search implements the browsing paradigm, representing a promising approach towards exploring and addressing (possibly fuzzy) information needs [13,5,11]. Here, users explore a collection of items by browsing conceptual dimensions of the items (i.e., facets) and their values (i.e., facet values) [10,21]. During an iterative process of selecting facets and refining the current result collection, users may construct complex, structured queries.

Related to faceted search is work on visual query builders [7,22]. However, while the latter focuses solely on intuitive means for query construction, faceted search aims at exploration and understanding (during a process of iterative query reformulation).

State-of-the-art. Faceted search was proposed for querying documents [10,9,4], databases [8,3,2] and semantic data [19,23,14] (referred to as *semantic faceted search*). One research direction is concerned with *efficiency* aspects. Existing work includes indexes and algorithms for fast computation of facets and facet-related data [4,9]. In this paper, we are concerned with the *effectiveness* of faceted search – efficiency aspects are orthogonal and unfortunately out of scope.

Given a large amount of facets associated with a collection of items, one major challenge we address is *facet ranking*. Widely used is *frequency-based ranking* [8,20,16]. It considers the number of items that are associated with a facet (its count). A facet is considered important, when its count is high. Based on the same idea, *set-cover ranking* has been proposed [8], which aims to maximise the number of distinct items that are accessible from the top-k facets. In [16], the authors assume a *relevance-based* ordering of items and propose ranking facets according to their likelihood of being associated with a relevant item. Further, the notion of *interestingness* has been incorporated into ranking [9], suggesting that facet relevance may be measured based on how surprising a facet is (given a certain expectation). The interestingness of a facet is defined as the aggregation of the interestingness of its facet values, which is based on rationales for what should be an expected facet value. In [8], the objective is to minimise *user costs* for finding a specific item of interest. Cost is defined as the time needed for reaching an item of interest. This time is computed as an aggregation of the times for reading facet headings, for browsing facet hierarchies and for correcting browsing mistakes. The authors of [3] propose to use the facet hierarchy (which the user traverses), as an approximation for the interaction time and cost, respectively. A ranking scheme is introduced to prefer facets with

a hierarchy of low height. Thus, facets are ranked high, when they quickly lead to an item of interest [3].

For effective faceted search, besides facet ranking, facet grouping approaches have been proposed. A *facet tree* (i.e., a tree-shaped facet grouping) was employed in [8,13,5,11]. Thus, users are able to browse multiple facets (forming a facet path), in order to explore a collection of items.

Contributions. We observe that (except for the generic, frequency-based ranking), existing ranking approaches assume a precise information need. That is, relevance, interestingness, and user costs (for fulfilling an information need) have been employed for measuring facet importance. Thus, we refer to such approaches as *search-oriented*. However, we propose a complementary approach, targeting a browsing scenario. Our solution supports users in addressing fuzzy needs, by enabling them to slowly explore an unknown collection of items. In particular, we provide mechanisms for grouping facets and facet values (i.e., an extended facet tree), as well as for ranking facets – both targeting an enhanced browsing experience. The contributions can be summarised as follows:

- For browsing facets with a large number of facet values, we propose an *extended facet tree*, which compactly captures both facets and facet values.
- We propose a ranking scheme, which supports users, given a fuzzy information need, in browsing a collection of items.
- We have implemented our approach and made the code[4] freely available. Further, we have conducted a task-based evaluation, showing that our approach outperforms the state-of-the-art on fuzzy information needs.

Outline. In Section 2, we introduce the data, query and facet model. Section 3 discusses (large) facet value sets and an extended facet model for browsing such sets. Facet ranking is discussed in Section 4. In Section 5, we present a task-based evaluation. We conclude with Section 6.

2 Data, Query and Facet Model

Data and Query Model. As different types of structured Web data may be represented as graphs (including RDF), we employ a general graph-structured data model [24].

Definition 1 (Data Graph). *Let $\mathcal{L}_V$ and $\mathcal{L}_E$ be finite sets of vertex and edge labels respectively. A data graph is a tuple $\mathcal{G} = (\mathcal{V}^G, \mathcal{E}^G)$, where $\mathcal{V}^G$ is a finite set of vertices, $l_V : \mathcal{V}^G \mapsto \mathcal{L}_V$ is a vertex labelling function and $\mathcal{E}^G \subseteq \mathcal{L}_E \times l_V(\mathcal{V}^G) \times l_V(\mathcal{V}^G)$ is a set of labelled edges. The set of vertices is conceived as the disjoint union $\mathcal{V}^G = \mathcal{V}_E^G \uplus \mathcal{V}_D^G$, where $\mathcal{V}_E^G$ stands for* entities *and $\mathcal{V}_D^G$ are data values. We distinguish the set of* relation edges $\mathcal{E}_R^G = \{e(v_i, v_j) \in \mathcal{E}^G | v_i, v_j \in \mathcal{V}_E^G\}$ *from the set of* attribute edges $\mathcal{E}_A^G = \{e(v_i, v_j) \in \mathcal{E}^G | v_i \in \mathcal{V}_E^G, v_j \in \mathcal{V}_D^G\}$ *and $\mathcal{E}^G = \mathcal{E}_R^G \uplus \mathcal{E}_A^G$.*

[4] http://code.google.com/p/semanticfacetedsearch/

Information needs in our setting correspond to conjunctive queries of the form $(x_1, \ldots, x_k).\exists x_{k+1}, \ldots, x_m.e_1 \wedge \ldots \wedge e_r$, where e_i are atomic formulae and $x_1, \ldots, x_k$ and $x_{k+1}, \ldots, x_m$ are called *distinguished* and *undistinguished variables*, respectively [24]. We focus on conjunctive queries with atomic formulae of the form $e(v_i, v_j)$, where v_i and v_j are either *variables* or *constants*. Since variables of a conjunctive query may interact in an arbitrary way, these formulae form a graph (so called *basic graph patterns*, representing a core feature of SPARQL). Further, a conjunctive query is denoted as $\mathcal{Q} = (\mathcal{V}^Q, \mathcal{E}^Q)$. Vertices of $\mathcal{Q}$ are $\mathcal{V}^Q = \mathcal{V}_V^Q \uplus \mathcal{V}_C^Q$ comprising a set of variables $\mathcal{V}_V^Q$ and constants $\mathcal{V}_C^Q \subseteq \mathcal{V}^G$. Edges of $\mathcal{Q}$ (called query predicates) are formulae $e(v_i, v_j)$, with $v_i \in \mathcal{V}_V^Q, v_j \in \mathcal{V}^Q$. A conjunctive query $\mathcal{Q}$ is processed as a graph pattern. Specifically, a result to $\mathcal{Q}$ on a graph $\mathcal{G}$ is a mapping from vertices of $\mathcal{Q}$ to vertices of $\mathcal{G}$, such that the substitution of variables (called variable bindings, denoted by $\mathcal{V}_x^R$ with $x \in \mathcal{V}_V^Q$) in $\mathcal{Q}$ would yield a subgraph of $\mathcal{G}$. Thus, every result represents a subgraph of $\mathcal{G}$. In fact, the entire set of results is a subgraph of $\mathcal{G}$, denoted by $\mathcal{R}(\mathcal{V}^R, \mathcal{E}^R)$ (called result set) [24].

Facet Model. Our conjunctive query model indicates the information needs we aim to support. However, via faceted search, users do not directly operate on this query model, but employ facets to construct queries [10]. With semantic faceted search [12,1,19,11], conjunctive queries may be constructed. To formalise the ideas of semantic faceted search, we employ a facet model comprising three components: (1) facets, (2) facet values and (3) facet operations.

Definition 2 (Facets). *Let $\mathcal{Q}(\mathcal{V}^Q, \mathcal{E}^Q)$ be the query, $\mathcal{R}(\mathcal{V}^R, \mathcal{E}^R)$ be the result set for $\mathcal{Q}$ and $\mathcal{V}_x^R \subseteq \mathcal{V}^R$ be the particular set of bindings obtained for the variable $x \in \mathcal{V}_V^Q$. Facets $F(x)$ (for the variable x) are labels of edges, which capture direct connections between elements in $\mathcal{V}_x^R$ and other elements of the data graph, i.e., $F(x) = \{f | f(v_i, v_j) \in \mathcal{E}^G, v_i \in \mathcal{V}_x^R\}$. Facets can be associated with every variable $x \in \mathcal{V}_V^Q$. The set of facets for $\mathcal{Q}$ is $F(\mathcal{Q}) = \{F(x) | x \in \mathcal{V}_V^Q\}$.*

Facets can seen conceived as conceptual dimensions of some particular variable bindings. In particular, every facet $f \in F(x)$ corresponds either to a relation or an attribute edge label. Thus, values of f might be entities or data values:

Definition 3 (Facet Values). *Let $\mathcal{R}(\mathcal{V}^R, \mathcal{E}^R)$ be the result set and $\mathcal{V}_x^R \subseteq \mathcal{V}^R$ be the bindings for a query variable x, then the values of a facet $f \in F(x)$ are entities or data values that are directly connected to elements in $\mathcal{V}_x^R$ via f, i.e., $FV(f) = \{v_j | f(v_i, v_j) \in \mathcal{E}^G, v_i \in \mathcal{V}_x^R\}$.*

There are three operations on facets that can be used to construct queries, i.e., to modify the bindings of variables $\mathcal{V}_{var}^R$ and thus, to modify the overall result set $\mathcal{R}$. These operations are: (1) *focus selection*, (2) *refinement* and (3) *expansion*.

With focus selection, users can change the focus to the variable (and thereby the set of bindings) they wish to modify. For instance, changing focus from y to x means to focus on facets contained in $F(x)$ (i.e., to focus on the entity set $\mathcal{V}_x^R$) instead of $F(y)$ (the entity set $\mathcal{V}_y^R$). In technical terms, it means that in faceted search, we have only one distinguished variable, which during the process, can

be changed by the user to obtain different sets of results for refinement and modification.

Users can modify the set of bindings for the variable in focus by adding further query predicates. In particular, a refinement operation performed on a variable x (on the entity set $\mathcal{V}_x^R$, respectively) means adding a new query predicate $f(x, y)$, with $f \in F(x)$ and y as new variable. Instead of adding further query predicates, a refinement can also be performed by modifying an existing query predicate. Let $f \in F(x)$ be a facet corresponding to the query predicate $f(x, y)$ and let $FV(f)$ be its facet value set. Users can refine $\mathcal{V}_x^R$ by choosing a facet value $v \in FV(f)$, in order to obtain a subset of $\mathcal{V}_x^R$ containing only entities connected to v via f. This refinement operation (denoted by $(f : v)$) replaces y in $f(x, y)$ with a constant v. Analogously, users may expand a result set by removing a facet (i.e., removing a query predicate) or removing a facet value (i.e., replacing a constant with a variable).

3 Browsing-Oriented Facet and Facet Value Spaces

In this section, we propose an extension of our facet model – the notion of a facet tree. For a result set, the basic facet tree (FT) compactly represents the space of all facets [12,1,19,11], while our extended facet tree (FT_e) additionally also captures the space of all facet values.

Facet Tree (FT). First, let us define the basic facet tree. For that, we introduce a *browsing* operation, which allows users to explore the facet (and later also the facet value) space via navigation along facets. Analogous to the expansion and refinement operation, browsing consists of (multiple) facet selections. However, facets selected during browsing are not evaluated, i.e., the underlying query does not change and thus the result set is not modified. A sequence of browsing operations allows users to navigate from the result set to associated facet values, and via their facets, to facet values that are further away. Every such browsing sequence establishes a facet path. All possible facet paths, which may result from browsing, establish a tree of facets:

Definition 4 (Facet Tree). *Let $\mathcal{G}(\mathcal{V}^G, \mathcal{E}^G)$ be the data graph and $\mathcal{V}_x^R \subseteq \mathcal{V}^R$ be the binding set (for a query $\mathcal{Q}$) for $x \in \mathcal{V}_V^Q$, then the facet tree $FT(x)$ for x can be conceived as a set of possibly overlapping paths P. Each $p \in P$ is of the form $\langle \mathcal{V}_x^R, \ldots, \mathcal{V}_l^L \rangle$, connecting the root node $\mathcal{V}_x^R$ with a leaf node $\mathcal{V}_l^L$. While leaves $\mathcal{V}^L$ are sets of data values, every other set $\mathcal{V}_i \in p$ comprises entities. There is a path $p \in P$ if and only if we find $\mathcal{V}_x^R, \ldots, \mathcal{V}_l^L \subseteq \mathcal{V}^G$ and $(\exists v_1 \in \mathcal{V}_x^R, \ldots, \exists v_l \in \mathcal{V}_l^L)$. $\exists e_1 \in \mathcal{E}^G, \ldots, \exists e_l \in \mathcal{E}^G$. $e_1(v_1, v_2) \wedge \ldots \wedge e_l(v_{l-1}, v_l)$.*

A facet tree FT is derived from vertices and edges in the data graph. In particular, FT captures all entities and data values reachable from the result set via navigation along paths in the data graph. It can be constructed via breadth-first search from the result set (the root) to data values (the leaves). Note, in order to include an entity v with no outgoing edges, we add a new edge $e(v, l_V(v))$ (i.e., an edge pointing to its label).

In Fig. 1 we illustrate an exemplary tree for a result set containing four professors. From the professor entities, it is possible to navigate to their associated names, their universities and the university's age.

Note, our notion of a facet tree is slightly different from existing facet trees and graphs. In Parallax [13] and Tabulator [5], it is possible to traverse tree-like structures, while with gFacet [11] even graph structures can be explored. With such systems, users can navigate from one entity set to another by changing the focus, and refine (expand) it by adding (removing) facets. In effect, users navigate though the data. As opposed to that, we explicitly employ a browsing operation, allowing users to explore the data without modifying the underlying query or changing the focus. The user does not browse through the data, but though a compact description, the facet tree. Users only see facets (e.g., *works at*, *age* and *name*) and (in the extended facet tree) labels of facet value sets (e.g., $University$ or $[ann - paul]$). Thus, browsing is based on a compact intensional description, thereby allowing users to easily grasp the overall structure of the facet (facet value) space. In order modify the result set via facet paths, we introduce an extension of the refinement operation. Instead of adding one query predicate, users can now add a conjunction of query predicates that corresponds to a facet path $\langle e_1, \ldots, e_l \rangle$ in FT, selected by the user (e.g., $\langle works\ at, age \rangle : [70 - 300]$). Analogously, we allow an extended expansion that removes a conjunction of query predicates.

Extended Facet Tree (FT_e). Through browsing along the paths of a facet tree, users eventually will reach a set of data values. Given a large set of data values, users might be overwhelmed. Thus, current systems choose to present only few top-ranked facet values and cut off the rest [19,23,14]. However, selecting one single value requires users to have precise needs and knowledge regarding that facet, which is contrary to the browsing paradigm. We aim to enable users to explore and understand the entire facet value space.

Therefore, we apply a *divisive hierarchical clustering* technique [15]. Using our clustering technique, a set of facet values (i.e., data values) $FV(f)$ is recursively split, resulting in a hierarchy of data value clusters. Since clusters at the same level do not overlap, the clustering process amounts to a partitioning of $FV(f)$. In order to decide where to split a cluster, a measure of dissimilarity between sets of data values is required. We use a distance measure for capturing the dissimilarity between

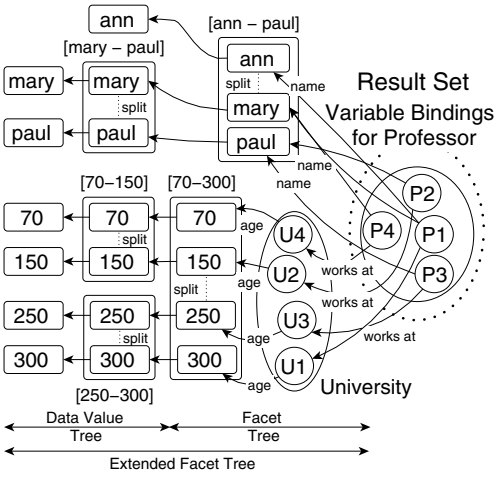


Fig. 1. Susan's (extended) facet tree

two data values and employ a linkage criterion, which captures the dissimilarity of sets as a function of the pairwise dissimilarity of data values (contained in those sets). More precisely, we use the Euclidean distance for numerical data values and the Levenshtein distance for textual data values. For computing the dissimilarity between sets of data values, we use the *single-linkage* criterion, where the distance between two sets of data values is defined as the minimum distance of all pairs of data values from both sets [15].

Now, we employ the clusters to extend the facet tree. Leaf nodes $\mathcal{V}_i^L \in FT(x)$ containing more data values than a given threshold are clustered, resulting in a set of *data value trees* (DT). The combination of facet tree and data value trees form an *extended facet tree* (FT_e). Note, clustering is performed only on demand (i.e., upon users browsing behaviour). More precisely, whenever a user reaches a facet tree leaf, associated facet values are clustered, and a data value tree is attached to extend FT.

Fig. 1 illustrates an exemplary extended facet tree. For instance, the set of names $\{ann, mary, paul\}$ is clustered, resulting in a tree of data values with $[ann - paul]$ as root. At the second level of the data tree, the set $[ann - paul]$ is split into two sets: $\{ann\}$ and $[mary - paul]$.

Compared to the state-of-the-art, the extended facet tree allows users to browse and explore the data based on a compact and hierarchical representation, using both facets (contained in FT) and facet value sets (contained in DT).

4 Browsing-Oriented Facet Ranking

Current work on facet ranking assumes users to have precise needs (i.e., know the domain and item of interest) and thus, relevance, interestingness, or user costs have been employed as measures [8,3,9]. In this section, we present a facet ranking scheme targeting a different scenario. We assume users to have incomplete knowledge w.r.t. domain or item of interest (i.e., fuzzy need). Thus, such users need support in exploring and understanding the result set. In particular, we prefer facets that allow users to modify the result set via small and uniform facet operations.

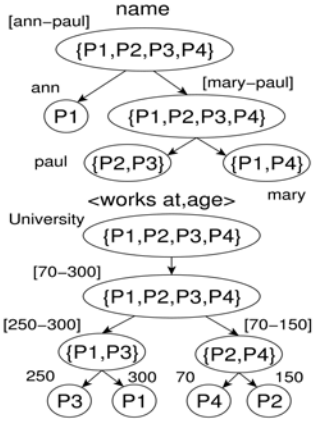


Fig. 2. Binding segments associated with facet *name* and *works-at*

4.1 Intuitions and Metrics for Browsing-Oriented Facet Ranking

For ranking a facet $f \in F(x)$, we consider the facet and facet value space that can be reached via f and result set modifications, which can be performed via facet paths originating from f.

More precisely, we consider f's extended facet tree: For facet f (representing the query predicate $f(x, y)$) we use $FT_e(y)$ to capture the facet and facet value

space reachable via f. That is, $FT_e(y)$ captures all facet paths that can be used to modify $\mathcal{V}_x^R$ ($\mathcal{V}^R$). We will now discuss the intuitions behind browsing-oriented facet ranking and concrete metrics used to measure them.

Small Steps. Users modify the result set until reaching an item of interest. Given that the facet paths (leading to an item of interest) are unknown and have yet to be explored, major result modifications that quickly change the result set are likely to lead to mistakes (i.e., lead to irrelevant results). Via small result modifications, users get to know the result set bit by bit. These small changes can be comprehended more easily by users (thus, they are less likely to choose paths that lead to irrelevant results). We use two metrics to implement this intuition:

- **Maximum Height (h).** The maximum height of FT_e (i.e., the maximum edge distance between the root node and a leaf node), directly reflects the maximum number of possible facet operations. Given the number of current results are fixed, the higher the number of possible result modifications, the smaller are the average changes resulting from each result modification. Thus, the greater the height of FT_e of f, the higher we rank f. In our example, the tree associated with *name* has height $h = 2$, while *works at* has a tree with $h = 3$. Thus, we prefer *works at* w.r.t. height.
- **Minimum Branching Factor (b).** The branching factor measures the number of nodes, which FT_e has at a particular level. Trees with small branching factor lead to smaller result modifications, as such trees tend to be higher. Further, a small branching factor reflects a small number of possible choices at every level in FT_e. Compared to a situation with a larger number of (necessarily more fine-grained) choices, this situation is easier for the user to cope with (as it does not require specific knowledge for making a decision). Therefore, we prefer facets having facet trees associated with a smaller branching factor. For instance, *name* and *works at* have the same rank in this regard because they have the same branching factor. At every level of both trees, Susan faces only two choices.

Uniform Steps. We consider query modifications to be non-uniform, when they have varying impacts on the result set size. More precisely, one query modification might strongly favour one particular segment of the result set and discriminate other segments. However, given the lack of precise knowledge about the item of interest, all results are a priori of equal importance. Thus, it is not possible to prefer a query modification that leads to a smaller set of results over another resulting in a larger set of results. Likewise, longer facet paths cannot be preferred over shorter paths. When browsing, it is hard for users to choose between these non-uniform query modifications. Such query modifications are rather confusing and likely lead to irrelevant results. Consequently, trees containing uniform query modification steps shall be preferred. We use metrics as follows:

- **Height Balance (hb).** FT_e is perfectly height balanced, when all leaves are of equal edge distance to the root. We define height balance as $hb(FT_e) =$

$\frac{c}{(dist_{max}(FT_e)+\epsilon)-dist_{min}(FT_e)}$, with c being a constant and $dist_{max}$ ($dist_{min}$) as maximal (minimal) edge distance from the root to a leaf. A facet with FT'_e associated is ranked higher than one with FT''_e, iff $hb(FT'_e) > hb(FT''_e)$. For instance, the facet tree associated with *name* is less height-balanced than the tree associated with *works at*.

– **Facet Value Set (sb_{fv}) and Binding Segment (sb_b) Size Balance.** We measure the size balance w.r.t. facet value sets and binding set segments, which may be reached via FT_e. Note that facet value sets are captured by leaf nodes $\mathcal{V}^L$ of FT_e. Let the leaf $\mathcal{V}^L_{max}$ ($\mathcal{V}^L_{min}$) contain the largest (smallest) number of facet values. The facet value set balance is $sb_{fv}(FT_e) = \frac{c}{(|\mathcal{V}^L_{max}|+\epsilon)-|\mathcal{V}^L_{min}|}$, with c being a constant. Further, every refinement operation (corresponding to a node $\mathcal{V}_i \in FT_e(x)$) actually leads to a binding segment $\mathcal{V}^R_{x_i} \subseteq \mathcal{V}^R_x$. Thus, corresponding to the facet tree, we have a tree of binding segments. Fig. 2 illustrates the trees of binding segments corresponding to the facet trees associated with *name* and *works at*. We consider the size of the binding segments at leaf level. For instance, *works at* has four binding segments at leaf level ($\{P1\}, \ldots, \{P4\}$); *name* has $\{P2, P3\}$ and $\{P1, P4\}$. Our variable binding segment size balance is $sb_b(FT_e) = \frac{c}{(|\mathcal{V}^A_{x_{max}}|+\epsilon)-|\mathcal{V}^A_{x_{min}}|}$, where c is a constant and $\mathcal{V}^A_{x_{max}}$ ($\mathcal{V}^A_{x_{min}}$) is the largest (smallest) variable binding segment (at leaf level). While the binding segment size is perfectly balanced for *works at*, this is not the case for *name* (one segment contains one professor, while the others contain two).

Comprehensible Result Segments. For users who are unfamiliar with a result set, it is important that a facet operation leads to obvious and comprehensible result modifications. Each operation should lead to a true result modification, i.e., an observable result refinement (expansion). Further, different operations should lead to different result modifications. Our metrics are:

– **Binding Distinguishability (d).** To assess whether facets lead to an observable result modification, we use the notion of distinguishability adopted from [3]. A facet has a high distinguishability, when it leads to facet values that precisely identify variable bindings. Ideally, leaves $\mathcal{V}^L$ are associated with binding segments consisting of exactly one element. Accordingly, our distinguishability metric is $d(FT_e) = \frac{|\mathcal{V}^R_x|}{\sum_{\mathcal{V}_i \in \mathcal{V}^L} |\mathcal{V}^R_{x_i}|}$. For instance, facet *name* leads to *mary*, which is associated with $\{P1, P4\}$. *Paul* is the name of $\{P2, P3\}$. Both pairs of professors share the same name, so it is difficult for Susan to distinguish them. Using the facet *works at*, she is able to distinguish between all four professors, as they work for different universities. Overall, *works at* has maximum distinguishability of 1, while the distinguishability for *name* is 0.8.

– **Minimal Binding Segment Overlap (o).** Binding segments with minimal overlaps are preferred to ensure that facet operations along a tree FT_e lead to different result modifications. Binding set overlap can be computed by considering the binding segment overlap at leaf level: $o(FT_e) = \frac{|\mathcal{V}^A_x|}{|\bigcap_{\mathcal{V}_i \in \mathcal{V}^L} \mathcal{V}^A_{x_i}|+\epsilon}$. In our example, the refinements ($name : ann$), ($name : mary$) and ($name :$

paul) split the binding set into segments that overlap on $\{P1\}$ (as $P1$'s name includes *ann* and *mary*). Refinements via $(\langle works\ at, age\rangle : 70)$ etc. split the binding set into segments with no overlaps. Thus, Susan can observe two sets of professors, i.e., one set $\{P2, P4\}$ working at older universities, while the other set $\{P1,P3\}$ is associated with younger universities.

4.2 A Browsing-Oriented Ranking Function

We now provide a scoring function $\mathcal{S}$, which incorporates the proposed metrics. We aim to rank a facet $f \in F(x)$, where f represents $f(x, y)$, based on its facet tree $FT_e(y)$. We distinguish facets that correspond to attributes from facets corresponding to relations.

Definition 5 (Attribute-based Scoring Function). *Given an attribute facet f, its facet tree FT_e, the score of f is defined as $\mathcal{S}(f) = ag(h(FT_e), b(FT_e), hb(FT_e), sb_{fv}(FT_e), sb_b(FT_e), d(FT_e), o(FT_e))$, with ag as a monotonic aggregation function.*

The set of attribute (relation) facets at level k that can be reached via the facet tree FT is denoted by $F_f^A(k)$ ($F_f^R(k)$). The score of a relation facet f at level k is computed based on the scores of facets $F_f^R(k+1)$ and $F_f^A(k+1)$.

Definition 6 (Relation-based Scoring Function). *Let f be a relation facet at depth $k = 0$ and let F_f^A (F_f^R) be the set of directly connected attribute (relation) facets (i.e., $F_f^A(1)$ and $F_f^R(1)$), k_{do} is the total edge distance to be considered, k_{did} is the edge distance considered so far, and $\delta(k)$ is a monotonic decreasing weight function discounting the score of facets more distant from f, then the score of f is recursively computed using the formula (starting at $k_{did} = 1$ and $k_{do} \geq 1$):*

$$\mathcal{S}(f) = \begin{cases} \delta(k_{did}) ag_{f_a \in F_f^A} \mathcal{S}(f_a) & if\ k_{do} = 1 \\ ag_{f_r \in F_f^R} \mathcal{S}(f_r)_{k_{do}-1, k_{did}+1} + \delta(k_{did}) ag_{f_a \in F_f^A} \mathcal{S}(f_a) & otherwise \end{cases}$$

In the current implementation, ag is a summation. We use $k_{do} = 1$, i.e., the score of a relation facet is simply an aggregation of the scores of reachable attribute facets.

5 Evaluation

We decided to conduct a task-based evaluation, which has gained acceptance in the IR community, especially for assessing approaches that go beyond IR-style document retrieval[5]. The goal is to find out whether browsing (as supported by our approach) helps to accomplish a set of predefined tasks (effectiveness) and how much time is needed (efficiency).

Participants. 24 participants took part in our evaluation: 6 were non-technical users, while the remaining 18 participants had a computer science background.

[5] http://trec.nist.gov/

All were familiar with faceted search (as used in Web search engines). The participants were given an introduction to the system, similar to an available screencast.[6]

Tasks. 24 tasks were chosen by domain experts and comprised both precise and fuzzy information needs. Tasks were followed by a series of questions to assess the user's understanding and exploration of the result set. A complete listing of tasks can be found in an extended technical report.[7]

Data. For the evaluation, we used the (complete) DBpedia dataset, which covers a large amount of broad-ranging knowledge [6]. DBpedia allowed us to design evaluation tasks that are not targeted at a particular domain.

System. We made use of the *Information Workbench*[6], a system for interacting with the Web of data. The proposed faceted search approach was implemented using Oracle Berkeley DB Java Edition and Apache Lucene, based on the design and indexes reported in [4,9]. We employed caching strategies to speed up cluster and rank computation and thereby guaranteed a fluent system interaction during the evaluation. Users were provided with a keyword search interface to obtain a starting point by typing in keywords. From the initial set of results, users continued via faceted search, i.e., via browsing, refinement and expansion operations. Results were visualised as introduced in [24], facets and facet values were presented as in [10], (extended) facet trees were represented as trees. Due to space reasons, we had to omit screenshots; however, they are included in our technical report[7]. The backend, including the keyword and faceted search modules, was implemented in Java 6 and the frontend is based on Ajax technologies running on a Jetty server. Experiments were carried out in a supervised manner on a PC with a T7300 Intel CPU and 4 GB memory, running on Microsoft Vista. We recorded the search process for each user and task using a screencast software.

5.1 Extended Facet Tree

Tasks. We prepared four tasks (C1-C4) for investigating the effects of our data value trees (i.e., data value clustering) on browsing. Task C3 is a precise need that involves a specific item of interest: *'Related to Berlin, find the Berlin Philharmonic orchestra'*. In contrast, the remaining 3 tasks involved fuzzy needs. In particular, tasks were fuzzy in the sense that the item of interest was specified imprecisely. Thus, participants had to browse and explore in order to fulfil these tasks. For instance, consider C2: *'Related to London, find all artists born some time in November 1972'*. Here, participants did not know what kind of artist or what concrete birthday to look for. The second class comprises eight complex browsing tasks (B1-B8). They have been designed to assess the quality of browsing based on the facet tree, compared to the baseline featuring a flat facet list. For accomplishing these tasks, users had to browse several facets to find a

[6] http://iwb.fluidops.net/
[7] http://www.aifb.kit.edu/web/Misc3004

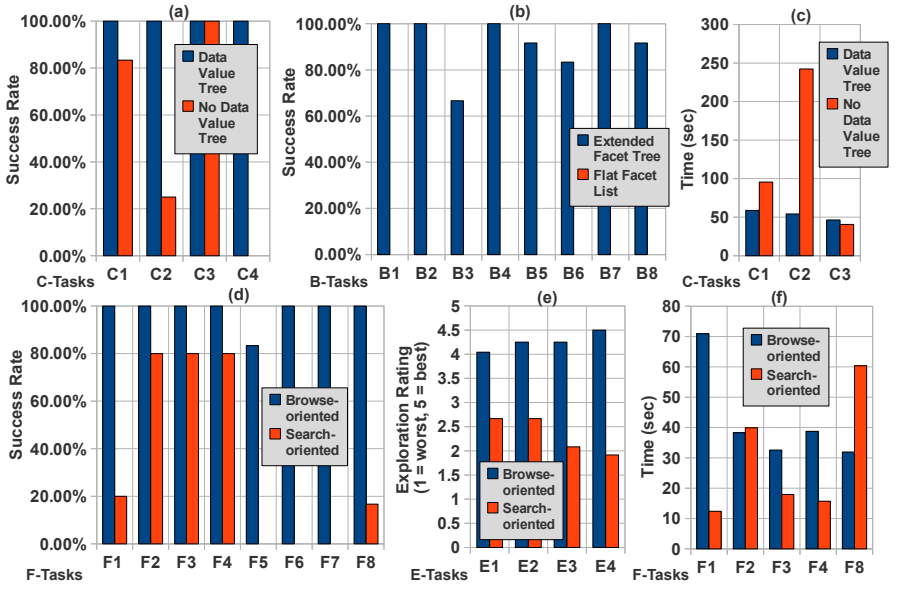


Fig. 3. Results of task-based evaluation

suitable facet path (length ≥ 2) for refinement (e.g., B1 *'Related to Paris, find all works having an actor, who is also a writer'*).

Baseline. In order to evaluate faceted search based on our extended facet tree, we used as baseline an implementation that represents the work in [23,1,19]. That is, browsing was solely based on a flat list of facets that are directly associated with the current result set. More precisely, we used two systems: (1) one system supported browsing on a flat list of facets without data value clustering and (2) our system that supported browsing based on the extended facet tree. Further, both systems employed a standard frequency-based ranking [8,20,16]. However, note, we designed clustering (C) and browsing (B) tasks in a way, that we were able to compare the effects of data value clustering en- or disabled and facets grouped in lists or trees. More precisely, in order to fulfil C tasks, users used made only use of a single facet and its associated data value tree or flat value list. Browsing tasks were designed analogously. Thus, we could observe the effects originating from facet trees versus lists and data value clustering versus no clustering.

Effectiveness. We observed that, if users were provided with a precise information need, the data value tree had no effect. Fig. 3a illustrates this result for task C3 – all participants could accomplish their assignment no matter the system. However, the fuzzier the information need, the more users depend on browsing the data value tree to solve their tasks (C1, C2, C4). More precisely, given a fuzzy need such as in C4 (*'Related to Hamburg, find all places having a name starting with [K-U]'*), we observed that without data value clustering, some users were not able or not willing to fulfil their assignment, because of the

substantial effort needed. With clustering enabled, participants achieved a high success rate, as shown in Fig. 3a. This result suggests that extending the facet tree with data value trees is helpful for browsing, enabling users to handle fuzzy needs via exploring both facets and facet values using a hierarchical structure.

Further, we observed that no participant managed to complete their browsing tasks using the baseline (featuring a flat facet list) (Fig. 3b). A number of users did realise that these complex tasks can solved by executing several searches and browsing several facet lists. However, they also noticed the substantial time required and were not willing to complete their tasks (e.g., B3: *'Related to London, find all works, having as subsequent work a television show'*). Using our system (providing the facet tree for browsing), participants achieved a high overall success rate of above 67% (Fig. 3b). This result is promising as all tasks involve complex information needs that can only be satisfied using complex structured queries. Participants solved them by exploring facet paths (often path length $\geq$ 3) and combining them in an iterative fashion.

Efficiency. Without data value trees and given fuzzy needs, we observed that, if participants completed their tasks, they had to use a brute-force strategy to search through a large set of facet values. The brute-force approach led to more system interactions and notably higher costs, when compared to our system (with data value trees). Fig. 3c shows this effect for C1 (*'Related to Paris, find all places, having names starting with [I-K]'*) and C2. Regarding tasks that involve precise information needs, many participants used search (based on keyword queries) as a strategy to complete their tasks (e.g., C3). This resulted in a performance comparable to the use of a data value tree. In fact, we observed our approach to be slightly more expensive in case of C3 (Fig. 3d), as the browsing operations performed on the data value tree took more time than search.

Concerning the complex browsing tasks (B1-B8), we already pointed out that no participant succeeded, when using the baseline (Fig. 3b).

In conclusion, the experimental results suggest that the use of a hierarchical facet model (like our extended facet tree) improves the efficiency and effectiveness of the task completion, concerning complex, fuzzy tasks. Search is more efficient and equally effective, with regard to precise and simple needs only.

5.2 Browsing-Oriented Ranking

Tasks. We prepared 12 tasks, which are divided into two classes: *find* (F) and *explore* (E) tasks. Class F consists of 8 tasks (F1-F8), which involve precise and fuzzy information needs (e.g., F8: *'Related to Seattle, find some international Airport'*). Class E (E1-E4) comprises 4 tasks, where users had to explore a result set (fuzzy need), i.e., find outliers, interesting or strange results (e.g., E4: *'Explore interesting entities related to Seattle'*). For E tasks, after a time threshold (5 minutes), users were asked a set of questions, in order to assess the users' understanding of the result set. Via these questions, users judged their understanding of the result set and rated the exploration experience and the knowledge that they could acquire on a scale from 1 (worst) to 5 (best).

Further, we divided the participants into two groups: For F tasks, the first group performed the tasks via search-oriented ranking, while the second group used browsing-oriented ranking. Thus, users solved each task only once. This is crucial, as the knowledge acquired from solving tasks using one system would impact on experiments with the other system. For E tasks, participants used both strategies, so that they could compare the exploration.

Baseline. Current ranking approaches are either generic w.r.t. information needs or assume precise needs. Our work is contrary to the latter approach and thus we compared our browsing-oriented ranking to such search-oriented approaches. More precisely, we used the metrics h and d (Section 4) and aggregated them to capture the intuition behind search-oriented ranking ($\overline{\mathcal{S}}$). Corresponding to the main idea (i.e., users having complete knowledge and therefore wish to minimise their search effort), $\overline{\mathcal{S}}(\hat{h}, \hat{d})$ aims at reducing the costs for fulfilling an information need, measured based on the number of refinement and expansion operations [8,3,9]. Thus, minimal tree height h is preferred to minimise the number of required facet operations. Further, facets should be discriminative, allowing users to quickly refine a result set. Thus, facets with high distinguishability score d are preferred. Overall, top-ranked facets aim at enabling users to perform rapid refinements and thereby reach an item of interest via few facet operations.

Effectiveness. For comparing the effectiveness of the two ranking strategies, we compared the average success rate for F tasks and the average browsing experience rating for E tasks. The results are depicted in Fig. 3d and Fig. 3e.

Concerning the success rate, given fuzzy needs, we observed that via search-oriented ranking, participants succeeded in tasks, where relatively small result sets (result size in the order of tens) had to be explored. Here, participants chose facets for browsing and refinement in a brute-force manner (e.g., F1 *'Related to Karlsruhe, find some city not located in Germany'*, or F3, F4 and F8). Concerning tasks with larger results (result size in the order of hundreds) and fuzzy needs, participants were not able to accomplish their assignments (e.g., F5 *'Related to Barcelona, find a strange educational institution'*, or F6 and F7).

Given precise needs, i.e., participants had precise background knowledge, users could solve their tasks equally effective with both rankings (e.g., for F2 *'Related to Karlsruhe, find a close-by airport'* some users knew that particular airport).

Browsing-oriented ranking outperforms the baseline on all tasks, especially those with large result sets and fuzzy needs. It seemed that for solving these tasks, users prefer general facets ranked high by the browsing-oriented strategy (e.g., *type* or *genre*), over fine-grained facets ranked high by the search-oriented strategy (e.g., *birthday* or *name*). As participants often had no precise knowledge regarding suitable values for such fine-grained facets (e.g., a specific birthday), they were not able to use them for exploration. Further, in many cases we observed participants using *type* for their initial exploration. *Type* helped them to get familiar with the current result set.

Concerning E tasks, the proposed ranking also performed well (Fig. 3e). Overall, exploration via browsing-oriented ranking was rated between 4 and 4.5, whereas exploration using search-oriented ranking was rated between 2 and 2.7.

Efficiency. For measuring user effort, we recorded the necessary time for all relevant system interactions, i.e., browsing, refinement and expansion operations. The time span for a browsing operation is defined as the time interval from the completion of the last facet operation, until the user browses to the next node in the facet tree (or decides to abort browsing). The time span for a refinement (expansion) operation is defined as the time from the last facet operation, until the user performs the next refinement (expansion).

On average, users needed 8 seconds for a refinement, 18 seconds for an expansion and 4.4 seconds for a browsing operation. The average time for each F task is illustrated in Fig. 3f. Note, F tasks that users did not complete for one of the systems are not shown (F5-F7).

Similar to the effectiveness study, we observed that the amount of results affects the performance of both strategies. When having a small result set (result size in the order of tens) and thus few facets, users solved their tasks on average with equal or less effort via search-oriented ranking (F1-F4). In particular for F2, users could exploit precise background knowledge (precise need), in order to refine the results set in an efficient, goal-directed manner. For the tasks F1, F3 and F4, users had few facets and thus were successful in guessing the appropriate facets that lead to their item of interest. Via browsing-oriented facets, on the other hand, more refinement and browsing operations were necessary, as high-ranked browsing-oriented facets restrict the result set in much smaller steps than search-oriented facets.

However, given fuzzy needs, when facing larger result sets (in the order of hundreds) and thus more facets, users were not able to guess suitable facets (F5-F8). More time had to be invested, as facet exploration was mere brute-force.

Thus, we can conclude that while browsing-oriented ranking might not provide the most efficient way to an item of interest, it is suitable for scenarios with no precise need and large result sets (thus, large facet and facet value spaces) to be explored.

6 Conclusion

Current faceted search approaches imply a precise information need and thus, focus on the search paradigm. We target the browsing paradigm, where users only vaguely know the domain or item of interest. To this end, we proposed the extended facet tree, which supports browsing based on a compact and hierarchical representation of the facet and facet value space. Based on the extended facet tree, we designed several metrics and incorporated them into a ranking scheme, which allows users to browse in small and uniform steps leading to observable and comprehensible result set modifications. We evaluated the proposed ranking and extended facet tree based on experiments with 24 tasks and 24 users. Our solution clearly outperformed the state-of-the-art on tasks, which involve fuzzy information needs and require dealing with large number of results. As

future work, we plan to integrate search- with browsing-oriented solutions, allowing varying types of information needs. In particular, we will study how to switch between search- and browsing-oriented ranking. Further, we will address efficiency aspects of the proposed ranking and facet tree computation.

References

1. Simile: Longwell rdf browser, http://simile.mit.edu/longwell/
2. Aditya, B., Bhalotia, G., Chakrabarti, S., Hulgeri, A., Nakhe, C., Parag, P., Sudarshan, S.: Banks: browsing and keyword searching in relational databases. In: VLDB, pp. 1083–1086 (2002)
3. Basu Roy, S., Wang, H., Das, G., Nambiar, U., Mohania, M.: Minimum-effort driven dynamic faceted search in structured databases. In: CIKM, pp. 13–22. ACM, New York (2008)
4. Ben-Yitzhak, O., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E., Sznajder, B., Yogev, S.: Beyond basic faceted search. In: WSDM, pp. 33–44. ACM, New York (2008)
5. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and analyzing linked data on the semantic web. In: Proceedings of the 3rd International Semantic Web User Interaction Workshop (2006)
6. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. Journal of Web Semantics 7(3), 154–165 (2009)
7. Catarci, T., Dongilli, P., Mascio, T.D., Franconi, E., Santucci, G., Tessaris, S.: An ontology based visual tool for query formulation support. In: ECAI, pp. 308–312 (2004)
8. Dakka, W., Ipeirotis, P.G., Wood, K.R.: Automatic construction of multifaceted browsing interfaces. In: CIKM, pp. 768–775. ACM, New York (2005)
9. Dash, D., Rao, J., Megiddo, N., Ailamaki, A., Lohman, G.: Dynamic faceted search for discovery-driven analysis. In: CIKM, pp. 3–12. ACM, New York (2008)
10. Hearst, M., Swearingen, K., Li, K., Yee, K.-P.: Faceted metadata for image search and browsing. In: CHI, pp. 401–408. ACM, New York (2003)
11. Heim, P., Ertl, T., Ziegler, J.: Facet graphs: Complex semantic querying made easy. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6088, pp. 288–302. Springer, Heidelberg (2010)
12. Hildebrand, M., van Ossenbruggen, J., Hardman, L.: /facet: A browser for heterogeneous semantic web repositories. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 272–285. Springer, Heidelberg (2006)
13. Huynh, D.F., Karger, D.R.: Parallax and companion: Set-based browsing for the data web. In: WWW (2009)
14. Hyvönen, E., Mkel, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., Kettula, S.: Museumfinland – finnish museums on the semantic web. Journal of Web Semantics 3(2), 25 (2005)
15. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
16. Koren, J., Zhang, Y., Liu, X.: Personalized interactive faceted search. In: WWW, pp. 477–486. ACM, New York (2008)

17. Marchionini, G.: Exploratory search: from finding to understanding. Commun. ACM 49(4), 41–46 (2006)
18. Marchionini, G., Shneiderman, B.: Finding facts vs. browsing knowledge in hypertext systems. Computer 21(1), 70–80 (1988)
19. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for rdf data. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 559–572. Springer, Heidelberg (2006)
20. Oren, E., Delbru, R., Möller, K., Völkel, M., Handschuh, S.: Annotation and navigation in semantic wikis. In: Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics (2006)
21. Ranganathan, S.R.: Colon Classification. Madras Library Association (1933)
22. Russell, A., Smart, P.: Nitelight: A graphical editor for sparql queries. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318. Springer, Heidelberg (2008)
23. Schraefel, Wilson, M., Russell, A., Smith, D.A.: mspace: improving information access to multimedia domains with multimodal exploratory search. Commun. ACM 49(4), 47–49 (2006)
24. Tran, T., Haase, P., Studer, R.: Semantic search — using graph-structured semantic models for supporting the search process. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009. LNCS, vol. 5545, pp. 48–65. Springer, Heidelberg (2009)

An Efficient Algorithm for Topic Ranking and Modeling Topic Evolution

Kumar Shubhankar, Aditya Pratap Singh, and Vikram Pudi

Center for Data Engineering, International Institute of Information Technology, Hyderabad, India
{shubankar,aditya_pratap}@students.iiit.ac.in, vikram@iiit.ac.in

Abstract. In this paper we introduce a novel and efficient approach to detect and rank topics in a large corpus of research papers. With rapidly growing size of academic literature, the problem of topic detection and topic ranking has become a challenging task. We present a unique approach that uses *closed frequent keyword*-set to form topics. We devise a modified *time independent PageRank* algorithm that assigns an authoritative score to each topic by considering the sub-graph in which the topic appears, producing a ranked list of topics. The use of citation network and the introduction of time invariance in the topic ranking algorithm reveal very interesting results. Our approach also provides a clustering technique for the research papers using topics as similarity measure. We extend our algorithms to study various aspects of topic evolution which gives interesting insight into trends in research areas over time. Our algorithms also detect hot topics and landmark topics over the years. We test our algorithms on the *DBLP* dataset and show that our algorithms are fast, effective and scalable.

Keywords: Closed Frequent Keyword-set, Topic Ranking, Citation Network, Authoritative Score, Evolution.

1 Introduction

The ever growing size of academic literature and fast changing fields of research pose a challenging task for a researcher to identify *significant* topics of research over the timeline. Topic discovery has recently attracted considerable research interest [13], [14], [15]. In this paper, we propose a novel and efficient method to detect and rank research topics. Based on the intuition that a document is well summarized by its *title* and the title gives a good high-level description of its content, we use the keywords present in the title of a paper to detect the topics. We form *closed frequent keyword-sets* as topics from the phrases present in the titles of papers on a user-defined *minimum support*.

We propose a time independent, modified iterative *PageRank* [3] algorithm to assign an authoritative score to the papers. For a topic *T*, we consider all the research papers containing that topic and the citation edges of these papers. We then assign an authoritative score to each topic using the scores of the papers containing that topic.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 320–330, 2011.

Our topic ranking algorithm is able to rank the topics based on their *significance* in research community rather than *popularity* of the topics, which only considers frequency of topics. All the papers sharing a topic form a natural cluster. It is to be noted that a paper could belong to a number of clusters forming *hierarchical*, *overlapping* clusters.

Considering the topics on year-wise granularity, we modeled the evolution of topics on timeline. We apply the evolution of the topics for First Topic Detection, finding Landmark Topics and Fading Topics. Our algorithms have many applications like topic recommendation systems for authors, trend analysis, topic search systems etc. We tested our algorithms on the *DBLP* dataset. Our experiments produced a ranked set of topics that on examination by field experts and based on our study match the prominent topics in the dataset over the timeline.

2 Related Work

Topic extraction from documents has been studied by many researchers. Most work on topic modeling is statistics-based like the work by Christain Wartena *et al.* [6], which use most frequent nouns, verbs and proper names as keywords. Our work is based on dissociation of phrases into frequent *keyword-sets,* which as discussed in section 2 is very fast and scalable. Topic summarization and analysis on academic documents has been studied by Xueyu Geng *et al.* [10]. They have used LDA model to extract topics which needs a pre-specified number of latent topics and manual topic labeling. In our study, no *prior* knowledge of topics is required. The work in [11] uses the correlation between the distribution of terms representing a topic and the links in the citation graph among the documents containing these terms. We have used frequent *keyword-sets* to form the topics and utilized the citation links to detect important topics among the topics derived.

Clustering documents based on frequent item-sets [1] has been studied in the algorithms FTC and HFTC [7] and the *Apriori*-based algorithm [8]. Both of these works consider the documents as bags of words and then find frequent item-sets. Thus, the semantic information present in the document is lost. We extract phrases from the titles of the research papers and derive its substrings as *keyword-sets,* maintaining the underlying semantics. We have used closed [2] frequent *keyword-set* rather than maximal frequent *keyword-set* as used by L. Zhuang *et al.* [9] in their work on document clustering. We cannot use maximal frequent *keyword-sets* as topics because then most of the information is lost as it considers only the longest possible *keyword-set*.

3 Topic Detection and Clustering

The method proposed by us is based on the formation of *keyword-sets* from titles of the research papers and finding closed frequent *keyword-sets* to form the *topics.*

Definition 1. Phrase: *A phrase P is defined as a run of words between two stop-words.*

Definition 2. Keyword-set: *A keyword-set K is defined as an n-gram substring of a phrase, n being a positive integer.*

Definition 3. Closed Frequent Keyword-set: *A keyword-set K is said to be frequent if its count in the corpus is greater than or equal to a user-defined minimum support [12].We define a closed frequent keyword-set as a frequent keyword-set none of whose supersets has the same cluster of research papers as it has.*

3.1 Phrase Extraction and Keyword-Set Formation

Given the title of a research paper R_i, we extract all its *phrases* $P_{ij,}$ where P_{ij} represents its j^{th} phrase. Each research paper R_i is mapped to the corresponding phrases P_{ij} present in its title. We reverse map the problem domain, mapping each phrase P_i to the research papers R_{ij} it belongs to, in one scan of the dataset. In this domain, each phrase will be dissociated into *keyword-set* only once, giving the frequency of *keyword-set* in the second scan.

In our approach, we have considered only the substrings of the phrases as *keyword-sets* and hence the relative ordering of keywords is maintained, preserving the underlying semantics. Each *keyword-set* thus formed is a semantic unit that can function as a basic building block of knowledge discovery and hence is a potential *topic.* As an example of *keyword-set* extraction, consider the phrase *xml data base,* the potential frequent *keyword-sets* are the set of all the ordered substrings giving the following *keyword-sets:* (*xml, data, base; xml data, data base; xml data base*). It is to be noted that finding all the substrings requires a simple implementation of queue in top-down fashion, taking $O(1)$ time at each level and $O(n)$ time overall. Deriving the substrings of a phrase rather than the power set of the keywords in the phrase which requires $O(n)$ time instead of $O(2^n)$.

3.2 Closed Frequent Keyword-Sets as Topics

Frequent keyword-sets are formed on a user-defined *minimum support*. The supports of the *keyword-sets* are calculated during the generation of the *keyword-sets* from the *phrases* in the second scan. The length of the list of research papers corresponding to a phrase is its *support*. It is to be noted that in the first scan, we cannot eliminate the phrases whose support is less than the minimum support as two or more phrases can share the same *keyword-set* whose combined support might be greater than the minimum support. The elimination of non-frequent *keyword-sets* is done only after all the *keyword-sets*, along with their supports, have been generated in the second scan. The algorithm to increment the support and add research papers to a given *keyword-set* is shown below:

```
Procedure 1: Frequent Keyword-set Generation
Require: phraseKeys PK, minimum support min_sup

1: for each phrase in PK
2:        keywordSetListKSL= findAllSubstringOf(P)
3:for each keywordSet K in KSL
4:        keywordSetCount[K] + = 1;
5:        add paper R to keywordSetPaperList[K]
6: for each keywordSet K in keywordSetCount
7:        if keywordSetCount[K] < min_sup
8:                delete(keywordSetCount[K])
9:                delete(keywordPaperList[K])
```

In the procedure *1*, all the frequent *keyword-sets* are derived along with their supports. From *step 1* to *step 5*, all the *keyword-sets* of each phrase are extracted and their supports in *keywordSetCount* and the corresponding paper list in *keywordSetPaperList* are updated. From *step 7* to *step 9*, we remove infrequent *keyword-sets*.

Traditional association rule mining algorithms like *Apriori* that require one scan of the dataset to calculate the supports of the item-sets at each level take too much time and space. In our algorithm, we require only *2* scans of the dataset to calculate the supports of *all* the candidate *keyword-sets*. Since our algorithm runs in *linear* time compared to *exponential Apriori* like algorithms, our algorithms are fast and highly scalable. Also, in *Apriori* like algorithms which build higher length item-sets from smaller ones, the relative ordering between the item-sets is lost. In our method, relative ordering of keywords is maintained preserving the underlying semantic of the phrases.

At this point, we have the frequent *keyword-sets.*. In our algorithm, we may derive non-closed frequent *keyword-set*s as well. Our topic should consist of the maximal number of common keywords present in all the papers in the cluster, so we remove the non-closed frequent keyword-sets. Thus, we have *closed frequent keyword-sets* as topics.

3.3 Clustering Research Papers Based on Topics

Till now, we have *closed frequent keyword-set*s as topics which act as the similarity measure to cluster the research papers. These topic clusters are complete in the sense that we have the maximal length *keyword-set* shared by all the papers represented by that topic. In the mapping *keywordSetPaperList*, corresponding to each topic, we have a list of papers, forming several hierarchical, overlapping clusters. The cluster representing a broader topic is essentially a combination of several clusters representing its sub-topics. For example, *databas* is a broad topic and *imag databas, distribut databas,* etc. are its sub-topics. Each level of the hierarchy represents a different level of data description, facilitating the knowledge discovery at various levels of abstraction.

4 Ranking of Topics

Our next step is to order the topics. At this stage, we have a comprehensive list of topics from various fields of research and on varied levels of abstraction. For a researcher looking for new topics for research, it becomes a very cumbersome task to go through the entire list of topics and decide upon which topics are *important.*

To determine the *importance* of a topic, we introduce an approach which is based on the intuition that the topic's *importance* should be determined by not only its frequency in the corpus but also the quality of papers in which the topic lies and quality of citations those papers have. To this end, our approach assigns authoritative scores [4] to the topics producing a ranked list of topics. For each topic we have a cluster of papers in which the topic lies. To find out which papers are of good quality, we have developed a time independent, modified *PageRank* algorithm using the citation network of the papers.

Definition 4. Citation Graph: *We define the citation graph G = (V, E) comprising a set V of nodes, which each node N_i representing a research paper R_i and a set E of directed edges, with each edge E_{ij} directed from the citing node N_i to the cited node N_j.*

Definition 5. Citation Sub-graph: *For a topic T, its citation sub-graph $G_T = (V_T, E_T)$ comprises the set V_T of nodes, where the topic T lies in each node and the edges citing these nodes(G_T can be collection of many sub-graphs not necessarily a connected-graph).*

Definition 6. Outlinks: *From a given node N, link all the nodes N_i that the node N cites.*

Definition 7. Inlinks: *To a given node N, link all the nodes N*j *that cite the node N.* The iterative formulae for calculating the *PageRank* score is:

$$PR(P) = (1-\theta) + \theta * \sum PR(P_i)/OC(P_i) \; . \tag{1}$$

Here $PR(P)$ is the *PageRank* score of the paper *P*. The *PageRank* algorithm is based on the fact that the quality of a node is equivalent to the summation of the qualities of the nodes that point to it. The *inlink* scores $PR(P_i)$ are divided by $OC(P_i)$ which is the number of *outlinks* of the *inlink* P_i. This takes care of the fact that if a paper cites more than one paper, it depicts that it has drawn inspiration from various sources and hence its effect on the score of the paper it cites should diminish by a factor equal to the number of paper it cites. The damping factor θ in the algorithm prevents the scores of research papers that do not have any *inlinks* from falling to zero. For the experiments we set the damping factor to *0.85* [3] which gave satisfactory results.

Time Invariant Factor: The basic *PageRank* algorithm does not take into consideration the time factor. It is observed that the newer papers do not get sufficient time to be cited compared to the older papers and thus fall behind in the ranking even if they are *important*. To counter this, we introduce a time-dependent metric which reduces the bias against the older papers to make the ranking time-independent. This metric Average Year Citations Count, *AYCC* is a time dependent metric and directly reflects the varying distribution of citations over the years. We observe that this metric captures the time bias against the newer papers well and has high values for older papers and low values for newer ones. It is calculated as:

$$AYCC(Y) = \sum(PI(P_Y))/N(P_Y). \tag{2}$$

$AYCC(Y)$ is the metric score for year *Y*. $PI(P_Y)$ is the inlink count for papers published in year *Y* and $N(P_Y)$ is the total number of papers published in the year *Y*. Considering the year of publication of all the research papers, we pre-compute the total number of citations for each year and the number of research papers published in each year. Using them, the average number of citations per paper for each year is determined. Its inclusion normalizes the biased distribution of citations on the timeline. We use this metric in calculation of the modified *PageRank* score by the following formulae:

$$MPR(P) = (1-\theta) + \theta * (\sum MPR(P_i)/OC(P_i))/AYCC(Y_P). \quad (3)$$

The *modified PageRank* score of paper *P, MPR*(*P*) incorporates the metric AYCC for its year of publication Y_P. Till now, we have *topic clusters T* each consisting of a number of research papers R_T dealing with research in the field represented by that topic. For ranking the topics, we use an authoritative score that takes into account the authoritative scores of the individual papers in the cluster. The formulae for topic score is as follows:

$$TS_T = (\sum MPR(P_T))/N_T. \quad (4)$$

The *topic score* TS_T of a topic *T* is the mean of the authoritative scores of all the research papers P_T present in the topic cluster, where N_T is the count of papers in the topic cluster *T*. Our algorithm is able to rank the topics based on their *significance*, considering the citation information as well as eliminating the time bias against the newer papers. Thus it is able to detect topics which may not be popular yet but may become popular afterwards. This information is not captured if we consider only the *frequency* of topics. For ex, if we analyze year-wise topics in the section *5.3* we find *mine associ rul* as top topic in *1993,* which shows that even if it had just emerged and had low frequency, it still was a *significant* topic due to important citations it received over the time.

5 Evolution of Topics

Every topic has its time-span. Topics evolve over time. It is important for a researcher to know how the topics are evolving, which topics are on the surge, which are on the decline and so on. Also, since we have the cluster of research papers corresponding to each topic and we have the scores of these papers, the papers with high scores can be labeled as the *important* papers of the corresponding topic. Topic evolution has following two notions:

- **Topic Year-wise:** We assign authoritative scores of all the papers in each topic to their year of publication and then calculated the average score of a topic for each year. This gives a clear idea how a topic has evolved over the years.
- **Year-wise Topics:** We assign each topic's scores for all the years as calculated above and for each year, sort the scores, taking only the top few topics. The results give a clear picture of how the top ranked topics vary over the years.

6 Experiments and Results

6.1 Dataset Description

To show the results of our algorithms, we used the *DBLP XML Records* [5] dataset. The *DBLP* dataset contains information about *1,632,442* research papers from various fields published over the years. It is to be noted that the dataset contained papers with citation information till the year *2010* only. As part of data pre-processing, the keywords present in the titles of the research papers were stemmed using the *Porter's* Stemming algorithm.

6.2 Results of Topic Ranking

An objective and quantitative evaluation of the results obtained is difficult due to the lack of standard formal measures for topic detection tasks. But, the ranked list of topics produced by our experiments on examination by field experts and based on our observations match the prevailing topics in the dataset. We tested our algorithms on various values of minimum support. Upon implementing the topic detection algorithms with minimum support *100*, we obtained *12,057* topics constituting *5,476* 1-length topics, *5,766* 2-length topics, *748* 3-length topics, *62* 4-length topics and *5* 5-length topics.

In the results, we show only those topics for which the number of papers in their cluster is more than a threshold η. This threshold is used so that the clusters suffice a minimum number of papers for authoritative score calculation. It should be noted that the threshold η considers only those papers in a cluster that have at-least one citation. The following table shows the top ten topics, where $\eta = 10$.

Table 1. Top *10* Topics with their Respective Authoritative Scores and Cluster Supports

Topic	Score	Support
congest avoid	*0.0791*	*112*
blind deconvolut	*0.0758*	*152*
learn tool	*0.0728*	*104*
sequenti process	*0.0719*	*112*
trecvid	*0.0716*	*111*
mine associ rule	*0.0710*	*197*
locat system	*0.0665*	*103*
hyperlink	*0.0662*	*200*
automat text	*0.0635*	*121*
large databas	*0.0623*	*346*

We see that the quality of a topic is dependent on both the quality of individual papers as well as the number of papers in the cluster. A topic with few but good quality papers can have a high ranking. Also, the topics that appear at the bottom of the ranking are the ones which have not been/could not be researched much. Some of such bottom-ranked topics are *radio access, ant coloni optim algorithm, x rai imag, ipv6 network,* etc. These topics can be of special interest to the new researchers looking for new dimensions of research.

6.3 Evolution of Topics

Topic Year-wise: The evolution of a topic is informative in itself. We can infer the birth of the topic, its period of *significant* impact and its end. Here, we present two graphs for the evolution of some selected topics showing their *average* and *cumulative* topic scores.

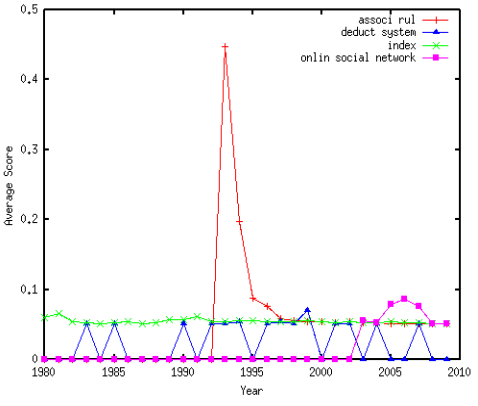


Fig. 1a. Graph showing average scores of the topics on year-wise granularity

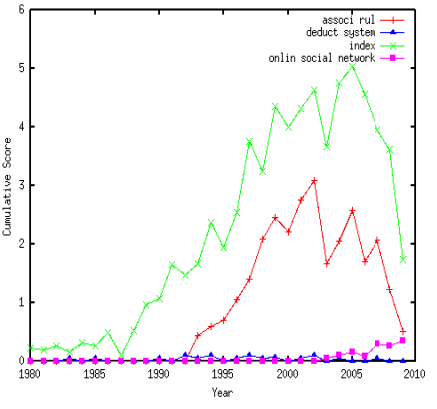


Fig. 1b. Graph showing cumulative scores of the topics on year-wise granularity

From the above two graphs, the following observations were obtained:

- If the average score of a topic is higher than that of another topic but its cumulative score is less, it means that the former topic has good quality papers in the cluster though few in number. For example, from *2005* to *2007*, the topic *onlin social network* has better average score than the topics *index* and *associ rul* but its cumulative score is less than both of them. This is because *onlin social network* was a new-born but *significant* topic, while *index* and *associ rul* were already known fields and considerable work was being done on them. Thus, both average and cumulative scores need to be considered to get a clear picture.
- The significance of the topic *associ rul* between *1993* and 1996 is clear as shown in Fig *1a* by its high average score which becomes similar to the score of *index* from *1997* onwards. From Fig *1b*, it can be seen that the cumulative score of *index* was always higher than that of *associ rul*. Thus after *1997*, the quality of *index* is similar to *associ rul*, but it was relatively more popular.

Another aspect of topic evolution could be studying the evolution of sub-topics of a topic. These sub-topics share a common *keyword-subset.* In the above graph in Fig 2, we show the evolution of the topic *databas* along with some of its sub-topics viz. *distribut databas*, *web databas* and *real tim databas*. As a topic consists of various sub-topics, at all points, some sub-topics lie above the topic graph while others below it. The topic score gets contribution from all its sub-topics in addition to its own topic cluster. Sub-topics like *distrbut databas* span a major part of timeline while some of the topics give way to other topics or evolve into other topics as is the case with *web databas* and *real tim databas*.

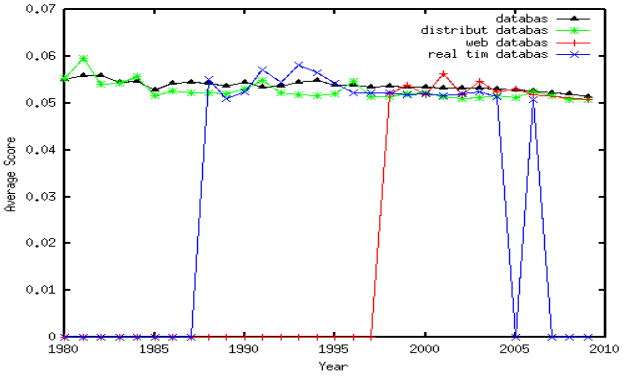


Fig. 2. Graph showing the evolution of *databas* and some of its sub-topics

Year-wise Top Topics: In this case, we compare all the topics for a given year. The following table shows the top three topics on year-wise granularity.

Table 2. Top Three Topics for each Year from 1993 to 2009

Year	Topic 1	Topic 2	Topic 3
1993	*mine associ rule*	*machin learn*	*larg databas*
1994	*associ rule mine*	*collabor filter*	*wordnet*
1995	*exchang blind deconvolut*	*data wareh environ*	*sequenti pattern*
1996	*data cluster*	*access control model*	*data hide*
1997	*adapt distribut*	*semi-structur data*	*collabor filter*
1998	*web search engine*	*wireless ad hoc network*	*anatomi*
1999	*learn tool*	*wireless sensor network*	*hyperlink*
2000	*instant messag*	*xml databas*	*evalu methodolog*
2001	*condit random field*	*dirichlet*	*peer to peer system*
2002	*stream system*	*cancer classif*	*k anonym*
2003	*transact memori*	*spatial correl*	*automat imag*
2004	*imag feature*	*network program*	*delaitoler network*
2005	*multi touch*	*object orient approach*	*onlin social network*
2006	*onlin social network*	*internet access*	*cyberspac*
2007	*evalu methodology*	*multimod interact*	*boltzmann machin*
2008	*distribut storage*	*trecvid*	*buffer manag*
2009	*fir filter*	*vision system*	*web portal*

Landmark Topics: We define *landmark topics* as those topics which gained extreme popularity within a short span of their emergence in the research domain. The year associated with a *landmark topic* is the year in which the topic *first* emerged. It is to be noted that these topics may not span sufficient number of papers but still our

time-independent modified *PageRank* algorithm is able to derive these topics. The following table shows the *landmark topics* that have emerged in the last fourteen years:

Table 3. Landmark Topics that Emerged Between 1996 And 2009

Year	Landmark Topics		
1996	*java*	*data cube*	*visual cryptographi*
1997	*xml*	*firewall*	*robocup*
1998	*web search engin*	*mobil ad hoc network*	*cellular neural network*
1999	*xml base*	*sensor network*	*dynam web*
2000	*mine frequent pattern*	*e busi*	*open sourc softwar*
2001	*multipath rout*	*intuitionist fuzzi set*	*multi hop wireless network*
2002	*pagerank*	*agil method*	*mobil learn*
2003	*gpu*	*microarray gene express*	*spam filter*
2004	*blog*	*bpel*	*multiplay onlin*
2005	*bit torr*	*cross layer design*	*3d face recognit*
2006	*cell broadband engin*	*folksonomi*	*web 2 0*
2007	*social media*	*ieee 802 16e*	*time delai system*
2008	*cuda*	*cloud comput*	*Svc*
2009	*microscopi imag*	*schrodinger equat*	*reson tunnel*

7 Conclusion and Future Works

In this paper, we proposed a method to derive topics, cluster papers into these topics and rank the topics using the authoritative scores of the constituent papers calculated by our time independent modified iterative *PageRank* algorithm. The topics were identified by forming *closed frequent keyword-sets* as proposed by our algorithms, which works better than traditional approaches like *Apriori.* We also studied the evolution of topics over time. We also analyzed the results of topic ranking and evolution of topics in detail.

As mentioned above, our algorithms have a variety of applications. In future, we would like to build topic recommendation systems. We would also like to examine statistical approaches for topic correlation and explore other domains like web site clustering, document clustering, etc. in which our algorithms can be applied.

References

1. Agarwal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. of the 20th VLDB Conference (1994)
2. Pasquier, N., Bastide, Y., Taoull, R., Lakhal, L.: Efficient Mining of Association Rules Using Closed Itemset Lattices. Information Systems (1999)
3. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: Proc. of the 7th International Conference on World Wide Web (1998)
4. Klienberg, J.: Authoritative sources in a hyperlinked environment. In: Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (1998)
5. The DBLP Computer Science Bibliography, `http://dblp.uni-trier.de/`

6. Wartena, C., Brussee, R.: Topic Detection by Clustering Keywords. In: Proc. of the 19th International Conference on Database and Expert Systems Applications (2008)
7. Beil, F., Ester, M., Xu, X.: Frequent Term-Based Text Clustering. In: Proc. of the 8th International Conference on Knowledge Discovery and Data Mining (2002)
8. Krishna, S.M., Bhavani, S.D.: An Efficient Approach for Text Clustering Based on Frequent Itemsets. European Journal of Scientific Research (2010)
9. Zhuang, L., Dai, H.: A Maximal Frequent Itemset Approach for Web Document Clustering. In: Proc. of the 4th International Conference on Computer and Information Technology (2004)
10. Geng, X., Wang, J.: Toward theme development analysis with topic clustering. In: Proc. of the 1st International Conference on Advanced Computer Theory and Engineering (2008)
11. Jo, Y., Lagoze, C., Giles, C.L.: Detecting Research Topics via the Correlation between the Graphs and Texts. In: Proc. of SIGKDD (2007)
12. Agarwal, R., Imielinski, T., Swami, A.: Mining Association Rules between Sets of Items in Large Databases. In: Proc. of the 1993 ACM SIGMOD Conference (1993)
13. Griffiths, T.I., Steyvers, M.: Finding Scientific Topics. Proc. of the National Academy of Sciences (2004)
14. Steyvers, M., Smyth, P., Rosen-Zvi, M., Griffiths, T.I.: Probabilistic Author-topic Models for Information Discovery. In: Proc. of SIGKDD (2004)
15. Mei, Q., Zhai, C.: Discovery Evolutionary Theme Patterns from Text – An Exploration of Temporal Text Mining. In: Proc. of SIGKDD (2005)

Sampling the National Deep Web

Denis Shestakov

Department of Media Technology,
Aalto University, Espoo, 02150 Finland
denis.shestakov@aalto.fi

Abstract. A huge portion of today's Web consists of web pages filled with information from myriads of online databases. This part of the Web, known as the deep Web, is to date relatively unexplored and even major characteristics such as number of searchable databases on the Web or databases' subject distribution are somewhat disputable. In this paper, we revisit a problem of deep Web characterization: how to estimate the total number of online databases on the Web? We propose the Host-IP clustering sampling method to address the drawbacks of existing approaches for deep Web characterization and report our findings based on the survey of Russian Web. Obtained estimates together with a proposed sampling technique could be useful for further studies to handle data in the deep Web.

Keywords: deep Web, web databases, web characterization, DNS load balancing, virtual hosting, Host-IP clustering, random sampling, national web domain.

1 Introduction

The deep Web, the huge part of the Web consisting of web pages accessible via web search forms (or search interfaces), is poorly crawled and thus invisible to current-day web search engines [13]. Though the problems with crawling dynamic web content hidden behind form-based search interfaces were evident as early as 2000 [6], the deep Web is still not adequately characterized and its key parameters (e.g., the total number of deep web sites and web databases, the overall size of the deep Web, the coverage of the deep Web by conventional search engines, etc.) can only be guessed.

Until now only a few efforts on the deep Web characterization have been done [6,9,16] and, more than that, one of these works is a white paper, with all findings obtained by using proprietary methods. Two other surveys, while being methodologically sound and reproducible, have inherent limitations due to the random sampling of IP addresses (rsIP for short) approach used in them. The most serious drawback of the rsIP method is the neglect of *virtual hosting*, i.e., a common practice of sharing one web server by multiple web sites. One of the largest European web hosting companies, OVH, with its 65,000 servers hosting over 7,500,000 sites[1] can be an example of actual ratios of web sites to servers

[1] See `http://www.ovh.co.uk/aboutus/ovh_figures.xml` (retrieved in May 2010).

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 331–340, 2011.

on the Web. The rsIP, however, usually detects from only one to three web sites per a server and ignores the rest, regardless of the actual number of sites on the server. In the context of deep Web characterization, ignoring the virtual hosting factor means that a portion of web sites (hosted on servers with sampled IP addresses) is overlooked, making the estimates obtained in the abovementioned surveys seriously biased.

Our contributions. We propose a novel method for sampling the deep Web, the Host-IP cluster sampling technique. Our approach is based on the idea of clustering hosts sharing the same IP addresses and analyzing "neighbors by IP" hosts together. Usage of host-IP mapping data allows us to address drawbacks of the rsIP used in previous deep Web surveys, specifically to take into account the virtual hosting factor. While we designed the proposed method for the survey of deep web resources on a national segment of the Web, it could also be applied to more general characterization studies of the entire Web.

Experimental results. To validate our technique, we applied the proposed approach to study the Russian segment of the deep Web. We used over 670,000 hostnames to generate around 80,000 groups of hosts which were then sampled, crawled and examined for the presence of search forms. Based on the results of sample analysis, we obtained statistically significant estimates for the total number of deep web sites and web databases in the Russian Web. We also compared our technique with the rsIP method and observed that the rsIP applied to the same data would result in substantial underestimations, e.g., would detect less than a third of deep web resources. Additionally, we demonstrated the magnitude of virtual hosting by reviewing previous web surveys (Section 2) and by resolving over 670,000 hostnames to their IP addresses (Section 5).

The next section gives a background on methods to characterize the deep Web. In Sections 3 and 4 we present our approach, the Host-IP cluster sampling technique. In Section 5 we report the results of the survey and compare the proposed method with the rsIP method. We then outline prior works and briefly discuss some of our findings in Section 6. Finally, Section 7 concludes the paper.

2 Background: Deep Web Characterization

Existing attempts to characterize the deep Web [6,9,16] are based on two methods originally applied to general Web surveys: namely, *overlap analysis* [7] and *random sampling of IP addresses* [12]. The first technique involves pairwise comparisons of listings of deep web sites, where the overlap between each two sources is used to estimate the size of the deep Web (specifically, total number of deep web sites) [6]. The critical requirement to listings be independent from one another is unfeasible in practice; making the estimates produced by overlap analysis seriously biased. Additionally, the method is generally non-reproducible.

Unlike the overlap analysis the second technique, the random sampling of IP addresses technique (rsIP), is easily reproducible and requires no pre-built listings. The rsIP estimates the total number of deep web sites by analyzing a

sample of unique IP (Internet Protocol) addresses randomly generated from the entire space of valid IPs and extrapolating the findings to the Web at large. Since the entire IP space is of finite size and every web site is hosted on one or several web servers, each with an IP address[2], analyzing an IP sample of adequate size can provide reliable estimates for the characteristics of the Web in question. In [9], one million unique randomly-selected IP addresses were scanned for active web servers by making an HTTP connection to each IP. Detected web servers were exhaustively crawled and those hosting *deep web sites* (defined as web sites with search interfaces, or search forms, that allow a user to search in underlying databases) were identified and counted.

Unfortunately the rsIP approach has several limitations. The most serious drawback is ignoring virtual hosting, i.e., the fact that multiple web sites can share the same IP address. This leads to ignoring a certain number of sites, some of which are apparently deep web sites. The reverse IP procedure (applied to obtain a hostname based on an IP address) typically identifies one or two web sites hosted on a given IP, while hosting a lot of sites on the same IP is a common practice. As an example, according to [1], 4.4 millions of IP addresses hosted almost 50 millions of hosts in April 2004 or, on the level of national domains, 640 thousands of second-level domain names in .RU and .SU zones resolved to 68 thousands of IPs in March 2007 (see the survey at `http://www.rukv.ru/runet-2007.html`).

Another factor overlooked by the rsIP method is *DNS load balancing*, i.e., the assignment of multiple IP addresses to a single web site. For instance, Russian news site `newsru.com` mapped to three[3] IPs is three times more likely to appear in a sample of random IPs than a site with one assigned IP. Since the DNS load balancing is the most beneficial for popular and highly trafficked web sites we expect that the bias caused by the load balancing is less than the bias due to the virtual hosting. Indeed, according to the SecuritySpace's survey as of April 2004, only 4.7% of hosts had their names resolved to multiple IP addresses [2], while more than 90% of hosts shared the same IP with others [1].

To summarize, the virtual hosting cannot be ignored in any IP-based sampling survey. Next we propose a new sampling strategy to address these challenges.

3 Our Approach

Real-world web sites are hosted on several web servers, share their web servers with other sites, and are often accessible via multiple hostnames. Neglecting these issues makes estimates produced by IP-based or host-based sampling seriously biased.

The clue to a better sampling strategy lies in the fact that hostname aliases for a given web site are frequently mapped to the same IP address. In this way, given

[2] An IP address is not a unique identifier for a web server as a single server may use multiple IPs and, conversely, several servers can answer for the same IP.

[3] Here and hereafter if not otherwise indicated, resolved in May 2010.

a hostname resolved to some IP address, we can identify other hostnames potentially pointing to the same web content by checking other hostnames mapped to this IP. It is interesting to see here a strong resemblance to the virtual hosting problem, where all hosts sharing a given IP address have to be found. Assuming a large listing of hosts is available, we can acquire the knowledge about which hosts mapped to which IPs by resolving all hostnames in the listing to their corresponding IP addresses. Technically, such massive resolving of available hosts to their IPs is essentially a process of clustering hosts into groups, each including hosts sharing the same IP address. Grouping hosts with the same IPs together is quite natural because it is exactly what happens on the Web, where a web server serves requests only to those hosts that are mapped to a server's IP. Once the overall list of hosts is clustered by IPs we can apply a cluster sampling strategy, where an IP address is a *primary sampling unit* consisting of a cluster of *secondary sampling units*, hosts.

Our Host-IP approach addressing all the drawbacks described in the previous section consists of the following major steps:

- *Resolving, clustering and sampling:* resolve a large number of hosts relating to a studied web segment to their IP addresses, group hosts based on their IPs, and generate a sample of random IP addresses from a list of all resolved IPs.
- *Crawling:* for each sampled IP crawl hosts sharing a sampled IP to a predefined depth. While crawling new hosts (which are not in the initial main list) may be found: those mapped to a sampled IP are to be analyzed, others are analyzed if certain conditions met (see Section 4.2).
- *Deep web site identification:* Analyze all pages retrieved during the crawling step and detect those with search interfaces to databases.

One can notice the principal difference between a sample unit of the rsIP method and a sample unit of the Host-IP approach. While all sampling units (IPs) in the rsIP are fully identical among each other, a sample of units in the Host-IP is heterogeneous. Indeed, in the Host-IP method there is an associated cluster of hosts for every sampled IP, and these clusters vary in size. Therefore, it could be useful to stratify, i.e., divide the resolved list of IP addresses into several non-overlapping parts (strata) and then deal with each part independently. The reasoning behind such separation is a reasonable assumption that deep web sites are more likely to be found within groups of hosts of certain sizes. If so, it might be beneficial to study groups with a few hosts separately from groups including hundreds of hosts. Another support for stratification is in the fact that IP addresses referred to a large number of hosts are good indicators of server hosting spam web sites [10] and, hence, deep web sites are less likely to be found among such IPs. Yet another reason is to actually verify whether deep web sites are running on servers hosting only a few sites. Anyhow, we note that the stratification itself is only a supplemental step and can easily be omitted.

4 Host-IP Cluster Sampling

In this section we give the schematic description of the Host-IP cluster sampling technique. The detailed description can be found in [15].

4.1 Dataset Preparation and Sampling

The steps for dataset preparation, clustering and sampling are as follows:

1. Obtain a set of unique host-IP pairs by resolving an available set of hosts to their IP addresses.
2. Remove host-IP pairs with invalid IP addresses.
3. Divide a set of host-IP pairs into two subsets S and S^*: S, which is used for clustering at step 4, has exactly one host-IP pair for each host and S^* includes all remaining pairs. Such separation allows us to avoid dealing with the DNS load balancing factor at the next clustering and sampling steps.
4. Group host-IP pairs in S by IPs and (optionally) stratify groups obtained by their sizes. Host-IP pairs with the same IP form a group. As a result, we obtain N groups, where N is the number of unique IP addresses among the pairs in S. Denote a set of all unique IPs in S as I and a set of all hosts in S as H. The number of pairs in a group (or in other words the number of hosts sharing a given IP) defines the group size and can be used as the stratification parameter.
5. Randomly select n IPs from I or, if stratified, for each stratum randomly select n_k IPs from I_k.

Now obtained sample (or, if stratified, samples) of IPs can be processed according to the crawling strategy presented next.

4.2 Crawling Strategy

Each IP in a given sample is processed independently from other IPs in this sample. The steps of the algorithm to crawl hosts (secondary sampling units) associated with a sampled IP are:

1. For each sampled IP ip, $ip \in I(I_k)$ extract from $S(S_k)$ a set of hosts H_{ip} sharing ip.
2. Each host in H_{ip} is crawled to a predefined depth. Crawling (i.e., following links) is done selectively: a link leading to a host that belongs to $H \setminus H_{ip}$ is not followed to not violate the sampling procedure. All other links are followed. Since it is expected that H has no full coverage of the studied web segment, we pay special attention to hosts out of H. So, while crawling hosts in H_{ip}, we add all unknown hosts to a set of hosts H^u_{ip}, i.e., $H^u_{ip} \cap H = \emptyset$.
3. After completion of crawling H_{ip} we proceed to crawl all hosts in H^u_{ip} to a predefined depth. Similar to the previous step following links is selective. A link to a host h' is followed only if $h' \notin H \setminus H_{ip}$ and (h' is in $H^u_{ip} \cup H_{ip}$ or h' is a subdomain of one of the hosts in $H^u_{ip} \cup H_{ip}$ or h' is resolved to ip). Unlike step 2 unknown hosts are not collected anymore.

After last IP in the sample is crawled, all pages retrieved are inspected according to the identification process revealed in the next section.

4.3 Deep Web Site Identification

All pages retrieved during the crawling step are analyzed for the presence of search forms (interfaces to web databases). In order to consider just unique search forms, pages with duplicated forms are removed. At the start, we exclude pages without web forms and then, based on the methodology described in [14], pages with non-searchable forms (i.e., forms that are not interfaces to databases such as forms for site search, navigation, login, registration, subscription, polling, posting, etc.). Identified pages with search interfaces are then grouped by their web sites. Web sites with two or more search forms are additionally studied to determine how many web databases are actually accessible via a particular site.

4.4 Estimates for Total Number of Deep Web Sites and Databases

Let N_k and n_k represent the total and sampled numbers of IP addresses of the k-th stratum correspondingly and h_{ki} the number of hosts on the i-th IP ($1 \leq i \leq N_k$) of the k-th stratum. The total number of hosts in stratum k is $H_k = \sum_{i=1}^{N_k} h_{ki}$, and the number of analyzed (sampled) hosts in stratum k is $h_k = \sum_{i=1}^{n_k} h_{ki}$. Let s_{ki} (d_{ki}) denote the number of deep web sites (databases) detected among the hosts on the i-th IP of stratum k. $s_{ki}(d_{ki}) = 0$ if no deep web sites (databases) are detected, $s_{ki}(d_{ki}) > 0$ otherwise. Then, according to Chapter 12, p.116 of [17], the estimate for the total number of deep web sites (databases) in the k-th stratum $\widehat{S_k}$ ($\widehat{D_k}$) is: $\widehat{S_k} = \frac{H_k}{h_k} \sum_{i=1}^{n_k} s_{ki}$ $\left(\widehat{D_k} = \frac{H_k}{h_k} \sum_{i=1}^{n_k} d_{ki}\right)$.

The estimator of the variance of $\widehat{S_k}$ is given by

$$\widehat{var}(\widehat{S_k}) = \frac{n_k(N_k - n_k)H_k^2}{N_k(n_k - 1)h_k^2} \sum_{i=1}^{n_k} (s_{ki} - h_{ki}\frac{\sum_{i=1}^{n_k} s_{ki}}{h_k})^2 \quad (1)$$

The estimator of the variance of $\widehat{D_k}$ is identical to (1) except that all s_{ki} in the formula should be replaced with d_{ki}. The approximate 95% confidence interval for the total number of deep web sites (databases) in the k-th stratum is provided by $\widehat{S_k} \pm t\sqrt{\widehat{var}(\widehat{S_k})}$ $\left(\widehat{D_k} \pm t\sqrt{\widehat{var}(\widehat{D_k})}\right)$, where t is the upper 0.025 point of Student's t distribution with $n_k - 1$ degrees of freedom. Finally, the approximate 95% confidence interval for the total number of deep web sites (databases) in all strata is

$$\sum_{k=1}^{L} \widehat{S_k} \pm t\sqrt{\sum_{k=1}^{L} \widehat{var}(\widehat{S_k})} \quad \left(\sum_{k=1}^{L} \widehat{D_k} \pm t\sqrt{\sum_{k=1}^{L} \widehat{var}(\widehat{D_k})}\right), \quad (2)$$

where L is the number of strata.

5 Experiments

For our experiments conducted in September 2006 we used two lists of hostnames from datasets "Hostgraph" and "RU-hosts". We merged them into one list of

Table 1. Number of IP addresses and hosts (total and sampled) in each stratum

Strata	Num of IPs:	Num of sampled IPs:	Num of hosts:	Num of sampled hosts:
Stratum 1 (S1)	71486	964	112755	1490
Stratum 2 (S2)	5390	100	86829	1584
Stratum 3 (S3)	1860	11	472474	3163

unique hostnames. Next, following the methodology described in Section 4.1, we built the dataset for our survey. We resulted in 717,240 host-IP pairs formed by 672,058 unique hosts and 79,679 unique IP addresses. These numbers specifically show us that DNS load balancing has a modest influence – only 5.4% (36,349) of hosts are mapped to multiple IPs, while most hosts, 94.6% (635,709), are resolved to a single IP address. At the same time, the compiled dataset gives yet another support for the magnitude of virtual hosting: there are, on average, nine hosts per one IP address[4]. 77.2% (553,707) of all hosts in the dataset share their IPs with at least 20 other hosts.

After exclusion of 'redundant' host-IP pairs from the overall set (step 3 of Section 4.1), we left with 672,058 host-IP pairs (672,058 unique hosts on 78,736 unique IP addresses) in the main set S and with 45,182 host-IP pairs in the 'redundant' set S^*.

We then clustered 672,058 host-IP pairs by their IPs and, in a such manner, got 78,736 groups of pairs, each having from one to thousands of hosts. We formed three strata using the following stratification criteria: Stratum 1 (S1) included those host-IP pairs which IP addresses are each associated with seven or less hostnames, groups of size from 8 to 40 inclusive formed Stratum 2 (S2), and Stratum 3 (S3) combined groups with no less than 41 hosts in each. 8 and 41 were chosen to make S1 contain 90% of all IP addresses and to put 70% of all hosts into S3. Table 1 presents the numbers of IP addresses and hosts in each stratum. One can particularly observe that S3 comprises 70% (472,474) of all hosts and only 2% (1,860) of all IP addresses.

We randomly selected 964, 100 and 11 primary sampling units (IP addresses) from S1, S2 and S3 correspondingly. It resulted in 6,237 secondary units (hosts) in total to crawl (see also Table 1 for numbers across strata). Hosts of every sampled IP were crawled to depth three[5] as described in Section 4.2.

We calculated the estimates for the total numbers of deep web sites and databases and their corresponding confidence intervals according to the formulas given in Section 4.4. The final results are presented in Table 2. The '*Num of all*' column shows (in italic) the numbers of deep web sites and web databases that were actually detected in strata. However, not all of them were appeared to be Russian deep web sites. In particular, several sampled hosts in .RU were in fact redirects to non-Russian deep web resources. Another noticeable example in this category was `xxx.itep.ru`, which is one of the aliases for the Russian-mirror of

[4] A host resolved to multiple IPs is counted for each corresponding IP.
[5] Discussion on crawling depth value is given in [3].

Table 2. Approximate 95% confidence intervals for the total numbers of deep web sites (*dws*) and web databases (*dbs*) in each stratum and in the entire survey

Strata	**Num of all:**		**Num of Russian:**		**Num of Russian, corrected:**	
	dws	dbs	dws	dbs	dws	dbs
Stratum 1:						
- *Detected in sample*	*80*	*131*	*72*	*106*	*61.2*	*86.7*
- Conf. interval, [10^3]	6.0±1.4	9.9±3.6	5.4±1.3	8.0±2.8	4.6±1.2	6.6±2.1
Stratum 2:						
- *Detected in sample*	*38*	*46*	*38*	*46*	*36.1*	*44.1*
- Conf. interval, [10^3]	2.1± 0.7	2.5± 1.1	2.1± 0.7	2.5± 1.1	2.0± 0.7	2.4± 1.1
Stratum 3:						
- *Detected in sample*	*64*	*87*	*55*	*68*	*51.2*	*62.6*
- Conf. interval, [10^3]	9.6±3.4	13.0±5.5	8.2±3.5	10.2±3.5	7.6±3.6	9.3±3.9
Survey total, [10^3]	17.7±3.7	25.4±6.5	15.7±3.7	20.7±4.4	**14.2±3.8**	**18.3±4.4**

arXiv (`http://arxiv.org/`), an essentially international open e-print archive. We excluded all such non-Russian resources and put the updated numbers in the '*Num of Russian*' column. We also examined each deep web site on its accessibility via host(-s) on IP(-s) different from a sampled IP (a corresponding weight should be assigned to a deep web resource accessible via hosts on two or more IPs [15]) and aggregated the numbers in the '*Num of Russian, corrected*' column of Table 2.

The survey results, the **overall numbers of deep web sites and web databases in the Russian segment of the Web as of September 2006** estimated by the Host-IP clustering method are **14,200±3,800** and **18,300± 4,400** correspondingly.

5.1 Comparison: Host-IP Clustering Method vs. rsIP Method

To compare the Host-IP method with the rsIP we used the list of Russian deep web sites detected by the Host-IP technique, namely, 72, 38 and 55 deep web sites found in samples of S1, S2 and S3 correspondingly (see the '*Num of Russian dws*' column in Table 2). We compiled the list of IP addresses on which these sites are running (multiple IPs were added for those mapped to multiple IPs) and then applied the rsIP method to the list. The results are summarized in Figure 1, where the left chart shows how many deep web sites within each specified group of sampled hosts were detected by the Host-IP and rsIP methods, and the right chart depicts the overall estimates produced by both methods for the numbers of deep web sites in each stratum and in total. For instance, the rsIP and Host-IP applied to hosts (of sample S1) sharing their IP with one or two other hosts detected 10 and 15 deep web sites correspondingly. The outcome is quite expectable while deep web sites of S3 were mostly undetectable by the rsIP method, around two thirds of deep web resources in S1 and one third of resources in S2 were successfully recognized by the rsIP. The interesting

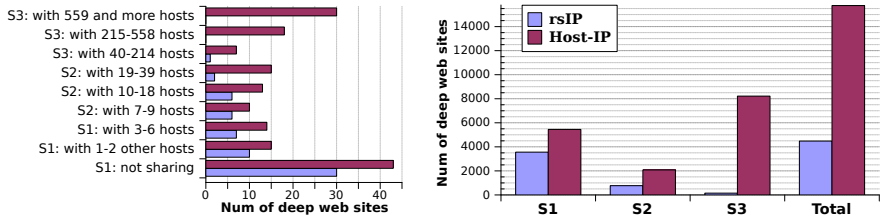


Fig. 1. Comparison of rsIP and Host-IP methods: *(left)* numbers of deep web sites detected in samples of strata (S1,S2,S3) among different types of hosts by rsIP and Host-IP; *(right)* numbers of deep web sites for each stratum and in total estimated by rsIP and Host-IP.

observation is that the rsIP is not efficient even for the hosts not sharing their IPs - 13 out of 43 deep web sites were overlooked by the rsIP[6] (see Figure 1(left)).

The factual estimates (approximate 95% confidence intervals) for the overall number of Russian deep web sites derived by the Host-IP and rsIP methods on the same IP list are 15,700±3,700 and 4,500±1,200 correspondingly. The rsIP approach therefore missed approximately seven out of ten deep web resources. In this way, **the rsIP** (used in previous deep Web characterization efforts) applied to our dataset **would be resulted in the estimates that are 3.5 times smaller than the actual figures**. The main impact to this difference is due to S3 with more than a half of all Russian deep web sites (8,200 out of 15,700), which are almost completely undetectable by the rsIP approach (see Figure 1(right)).

6 Related Work and Discussion

In Section 2, we mentioned existing deep Web surveys and discussed their limitations, where the most serious one is ignoring the virtual hosting factor. Several studies on the characterization of the indexable Web space of various national domains have been published (e.g., [5,11,18]). The review work [4] surveys several reports on national Web domains, discusses survey methodologies and presents a side-by-side comparison of their results. The idea of grouping hosts based on their IP addresses was used by Bharat et al. [8] to identify host aliases (or mirrored hosts according to Bharat's terminology). At the same time, we are unaware of any web survey study based on the Host-IP clustering approach.

One of the most surprising results in our survey is the fact that around a half of all deep web sites are hosted on IP addresses shared by more than 40 hosts (see 'S3' row of Table 2). It is somewhat unexpected since a deep web site serves dynamic content and thus normally requires more resources than an ordinary web site. Common sense suggests that a dedicated server (i.e., a server hosting from one or two to perhaps dozens of hosts, most of which are aliases) would be a better alternative for hosting a web site with database access. Nevertheless, it gives us just another strong justification for taking into consideration the virtual hosting factor.

[6] Empty results of reverse IP resolving were the reasons of overlooks.

7 Conclusions

We described a new sampling strategy, the Host-IP cluster sampling, that addresses drawbacks of previous deep Web surveys and accurately characterizes a large national web domain. We demonstrated the magnitude of virtual hosting and showed the consequences of ignoring it on a real dataset. We also compared our approach with the rsIP technique used in previous deep Web characterization studies and showed that the rsIP estimates for total number of deep web sites and databases are highly underestimated. Finally, we conducted the survey of Russian deep Web and estimated, as of September 2006, the overall number of deep web sites in the Russian segment of the Web as 14,200±3,800 and the overall number of web databases as 18,300±4,400.

References

1. April 2004 Web Server Survey (April 2004), http://news.netcraft.com/archives/2004/04/01/april_2004_web_server_surv%ey.html
2. DNS load balancing report (April 2004), http://www.securityspace.com/s_survey/data/man.200404/dnsmult.html
3. Baeza-Yates, R., Castillo, C.: Crawling the infinite Web: five levels are enough. In: Leonardi, S. (ed.) WAW 2004. LNCS, vol. 3243, pp. 156–167. Springer, Heidelberg (2004)
4. Baeza-Yates, R., Castillo, C., Efthimiadis, E.N.: Characterization of national Web domains. ACM Trans. Internet Technol. 7(2) (2007)
5. Baeza-Yates, R., Castillo, C., López, V.: Characteristics of the Web of Spain. Cybermetrics 9(1) (2005)
6. Bergman, M.: The deep Web: surfacing hidden value. Journal of Electronic Publishing 7(1) (2001)
7. Bharat, K., Broder, A.: A technique for measuring the relative size and overlap of public web search engines. Comput. Netw. ISDN Syst. 30(1-7), 379–388 (1998)
8. Bharat, K., Broder, A., Dean, J., Henzinger, M.: A comparison of techniques to find mirrored hosts on the WWW. J. Am. Soc. Inf. Sci. 51(12), 1114–1122 (2000)
9. Chang, K., He, B., Li, C., Patel, M., Zhang, Z.: Structured databases on the Web: observations and implications. SIGMOD Rec. 33(3), 61–70 (2004)
10. Fetterly, D., Manasse, M., Najork, M.: Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In: Proc. of WebDB 2004 (2004)
11. Gomes, D., Silva, M.J.: Characterizing a national community web. ACM Trans. Internet Technol. 5(3), 508–531 (2005)
12. O'Neill, E.T., McClain, P.D., Lavoie, B.F.: A methodology for sampling the World Wide Web. Annual Review of OCLC Research 1997 (1997)
13. Shestakov, D.: Deep Web: databases on the Web. In: Handbook of Research on Innovations in Database Technologies and Applications, pp. 581–588. IGI Global (2009)
14. Shestakov, D.: On building a search interface discovery system. In: Proceedings of VLDB Workshops 2009, pp. 114–125 (2009)
15. Shestakov, D.: Measuring the deep Web (2011) (submitted)
16. Shestakov, D., Salakoski, T.: On estimating the scale of national deep Web. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 780–789. Springer, Heidelberg (2007)
17. Thompson, S.: Sampling. John Wiley & Sons, New York (1992)
18. Tolosa, G., Bordignon, F., Baeza-Yates, R., Castillo, C.: Characterization of the Argentinian Web. Cybermetrics 11(1) (2007)

A Bipartite Graph Model and Mutually Reinforcing Analysis for Review Sites

Kazuki Tawaramoto, Junpei Kawamoto*, Yasuhito Asano, and Masatoshi Yoshikawa

Graduate School of Informatics, Kyoto University, Kyoto, Japan,
{tawara,j.kawamoto}@db.soc.i.kyoto-u.ac.jp,
{asano,yoshikawa}@i.kyoto-u.ac.jp

Abstract. A number of methods have been proposed for detecting spam reviews in order to obtain credible summaries. These methods, however, could not be uniformly applied to various forms of reviews and are not suitable for a product or service which has been evaluated by few reviewers. In this paper, we propose a bipartite graph model of review sites and a mutually reinforcing method of summarizing evaluations and detecting anomalous reviewers. Our model and method can be applied to reviews of various forms, and is suitable for a subject with few reviewers. We ascertain the effectiveness of our method using reviews of three forms on Yahoo! Movie web site.

Keywords: graph model, mutually reinforcing, anomally detection.

1 Introduction

There are a number of Web sites publish user reviews about various kinds of target objects including products or services. Reviews can be various forms such as binary evaluation (i.e. good or bad), a single score, a vector representing scores of multiple aspects, and sentences. For example, in Amazon.com[1], a reviewer can submit a single score and a few sentences to evaluate a product. Such review sites have been considered as an important tool to help both consumers and vendors to make decisions. A consumer can find a better product reading product reviews on Amazon.com and vendors can obtain feedback about their products and services from review sites in order to utilize that for marketing. A summary of reviews about a target object could be useful for making decisions particularly because the summary enables us to understand semantic orientations without reading all the reviews. In fact, most of review sites provide a summary of reviews. A typical summary is the average of scores assigned by reviewers.

In summarizing reviews about a target object treating all reviews equivalently is inappropriate approach for these methods, because some malicious reviewers would bring a biased view deliberately for their benefit and some reviewers would

* Research Fellow of the Japan Society for the Promotion of Science.
[1] http://www.amazon.com/

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 341–348, 2011.

provide too specialized opinions to be referred by common people. Such reviewers are called anomalous ones, while malicious reviewers are called spam reviewers. An example of anomalous but not spam reviewers is experts about a product. Their opinions are often significantly useful for another expert, although they can be inappropriate for common people. For example, if a common person desires to buy a digital camera for daily use, then reviews of common people are more suitable than those of experts; a digital camera recommended by experts is often too expensive for common people. Therefore, detecting anomalous reviews or reviewers is useful for summarizing reviews. A number of methods have been proposed for detecting anomalous reviews or reviewers.[4,1,5] However, most of these methods have the following three problems.

Problem 1: These methods have been developed independently of methods for summarizing reviews; consequently, they could not fully utilize the relationship between anomalous reviewers and the summary of reviews.
Problem 2: Ideas used for detecting anomalous reviewers depend on each of forms of reviews; it is desired that an idea applicable to all the forms uniformly.
Problem 3: These methods requires a large amount of training data for detecting anomalous reviewers if a target object has few reviews.

Problem 3 is important for a new product or movie because the people experienced them might be very few and old training data might be useless due to completely new topics in them.

In this paper, we propose a novel model and method in order to dissolve the three problems. First, our model is a bipartite graph model of reviewers, products or services, and evaluations. Evaluations are allowed to be quantified in any forms and tied to edges in our model. In addition, a bipartite graph model can treat all evaluations of a reviewer or all evaluations to a target object with its adjacency. Thus, this model is suitable to treat both detecting anomalous reviewers and summarizing reviews uniformly. Second, our method is a mutually reinforcing method of summarizing reviews and detecting reviewers. In the proposed method, we use a mutually reinforcing relation which is the relation that two properties reinforce each other by the supports: what has property a is supported by what has property b and what has property b supports what has property a. We name the analysis using this relation as a mutually reinforcing analysis. Using a mutually reinforcing relation, we can treat both detecting anomalous reviewers and summarizing reviews uniformly. HITS [3], the method of discovering authoritative web pages (authorities) and the pages containing many links to them (hubs), is regarded as a mutually reinforcing analysis because links represent supports and there is a mutually reinforcing relation between authorities and hubs: a good authority is linked by many good hubs and a good hub links to many good authorities. Existing mutually reinforcing analyses express each property in real numbers. However, evaluations can be also quantified in more intricate forms, such as vectors and distributions. Thus, we propose a mutually reinforcing method which has applicability to any forms of quantified evaluations. Additionally, our method is unsupervised and need no training data. Even

if a target object has few reviews, our method can detect anomalous reviewers using both reviews in a target object and those in other objects.

We ascertain the effectiveness of our method using reviews of three form and the situations in which there are some objects reviewed by few reviewers. In the result, our method can detect anomalous reviewers and create more credible summary than the average of reviews even if the target object has few reviewers.

2 Related Work

Our detection of anomalous reviewers is included in anomaly detection. While we described in Section 3, we use contextual anomalies to detect anomalous reviewers. The context we use is a reviewed object and we detect anomalous reviewers among those giving a review of the object. Some existing studies have examined contextual anomaly detection on a graph modeling target data. In a bipartite graph, Sun et. al [7] define the similarity of nodes and detect anomalous nodes among those which are adjacent to a given node. Wang et. al [8] detects a context and anomalies in that context in data set. While they use the similarity of nodes for detection, we use quantified evaluations tied to edges. In addition, each method use only real numbers in detection, but our method can use other forms of feature amounts, such as a vector and a distribution.

Some existing studies define spam reviews or spam reviewers and detect those reviews or reviewers. Lim et. al [4] propose four models based on their behavior of rating evaluations and reviews and detect spam reviewers using them. Jindal et. al [1] classify spam reviews into three types, and detect each type of spam using various models. Liu et. al [5] define the quality of product reviews and detect those of low quality. They also apply the proposed detection method as a filtering method to refine a summary of product reviews. Anomalous reviewers we detect include spam reviewers. In addition, we detect reviewers who evaluate differently from common people such as experts of reviewed objects. We consider it suitable that such reviews have a less power in a summary of reviews for common people.

3 Proposed Method

3.1 Bipartite Graph Model for Review Analysis

Our graph model for review analysis has two kinds of instances; subject and object. A subject belongs to a group in which instance gives evaluations. In typical review sites, reviewers are included in this group. On the other hands, an object belongs to a group in which instance is evaluated. For example, products and services are included in this group. For summarizing evaluations and detecting anomalous subjects, we assume that each evaluation is given to one object by one subject. On this assumption, we can model review data as a bipartite graph $G(V_S, V_O, E)$ which can express those subset of evaluations with its adjacency. The notation of V_S, V_O, E are defined as shown below:

$\mathbf{V_S} = \{p\}$: set of the nodes which represent subjects,
$\mathbf{V_O} = \{q\}$: set of the nodes which represent objects,
$\mathbf{E} = \{(p, q)\}$: set of the edges which represent evaluations.

In those expressions, (p, q) is generated and tied to quantified evaluation D_{pq} if p evaluates q. D_{pq} is allowed to be any forms, such as a real number, a vector, and a distribution expect that D_{pq} is the same form in $\forall p$ and $\forall q$.

3.2 Mutually Reinforcing Analysis

On the graph model explained in Section 3.1 we summarize evaluations and detect anomalous subjects uniformaly. When we analyze a graph model, we give each node of V_S and V_O a feature amount based on a mutually reinforcing relation among subjects and objects, as described below.

- an anomalous subject gives an extremely different evaluation from a summary of evaluations to each of many objects,
- a summary of evaluations is similar to evaluations of many normal subjects.

Based on that mutually reinforcing relation, we give a subject p a feature amount x_p of real number representing the degree of anomalous subjects and give an object q a feature amount Y_q of the same form of quantified evaluations D_{pq} representing a summary of evaluations. For example, when a evaluation expresses a positive or negative polarity and is quantified as a real number D_{pq}, Y_q is a real number, too.

We assume that the degree of anomalous subjects of p is determined by dissimilarities between D_{pq} and Y_q in $\forall q$ which is adjacent to p in a graph model. So, we define x_p as

$$x_p = \sum_{q:(p,q)\in E} \frac{x_{pq}}{N_p},\ x_{pq} = distance(D_{pq}, Y_q),$$

where N_p is the number of nodes which are adjacent to p. x_{pq} is the dissimilarity between D_{pq} and Y_q given by a specified distance function *distance*. Selecting a *distance* depending on the form of D_{pq} and Y_q, we can apply formula 3.2 to any forms of D_{pq} and Y_q. x_{pq} means the degree of anomaly of the evaluations of p to q. x_p is calculated by integrating x_{pq} in $\forall q$ which is adjacent to p. In this time, we integrate all x_{pq} to average them.

We also assume two things: (1)a summary of evaluations to q is determined by both x_p and D_{pq} in $\forall p$ which is adjacent to q in a graph model; (2)the more anomalous subjects effect the less for a summary of evaluations. So, we define Y_q as follows,

$$Y_q = \sum_{p:(p,q)\in E} w_{pq} D_{pq},\ w_{pq} = \frac{\frac{1}{x_p}}{\sum_{p:(p,q)\in E} \frac{1}{x_p}},$$

where w_{pq} is used in the weighted average for a summary of evaluations of q. w_{pq} is calculated based on x_p and $\sum_{p:(p,q)\in E} w_{pq} = 1$. Y_q is the weighted average using w_{pq} and D_{pq}.

3.3 Computing Feature Amounts

To compute feature amounts x_p and Y_q, we use an iterative algorithm. First, we initialize $\forall x_p$ with $x_p^{<0>}$ where $x_p^{<0>}$ is the same value in $\forall p$. Then, we update both the feature amounts using above definitions in iterative fashion until they converge. After they converged, we obtain feature amounts from converged amounts of x_p and Y_q. However that algorithm is one which presumes the convergence of feature amounts, the convergence is not guaranteed qualitatively. Proving the convergence condition is our future work. Therefore, we continue to update both the feature amounts until the difference of each x_p is sufficiently small. Then, we finish it.

4 Experiment

4.1 Dataset

To ascertain the effectiveness of our proposed method, we used user reviews posted on Yahoo! Movie web site[2]. We think movies which have just run in theaters or which are minor can be evaluated by fewer reviewers and anomalous evaluations can have a power on a summary of evaluations in such situations. Thus, to create such a small dataset, we selected 17 reviewers from original data and created three anomalous reviewers who provided random evaluations for randomly selected movie titles. In this experiment, we used both the dataset including only original 17 reviewers and the dataset including three anomalous reviewers and original 17 reviewers.

We propose a method which has applicability to any forms of quantified evaluations, so we use three forms of quantified evaluations, as described below.

Evaluation score (E). We used the rating score: *saiten* (grade) in user reviews. It is a real number of 1-5.

Feature-based evaluation vector (FE). We also used other rating scores: *monogatari* (story), *haiyaku* (casting), *ensyutsu* (direction), *eizo* (visuals), *ongaku* (music) in user reviews, and created five-dimensional vectors. Each element of those vectors is a real number of 1-5.

Sentiment model-based evaluation vector (S). We extracted and quantified sentiments from texts in user reviews using a sentiment dictionary [2]. In [2], they use the sentiment model proposed by Plutchik [6]. This model has 8 primary emotions: *acceptance*, *joy*, *anticipation*, *anger*, *disgust*, *sadness*, *surprise*, and *fear* and represents all emotions using the combination of some primary emotions. In this model, each primary emotion has its opposite primary emotion and a pair of one and its opposite can make an axis: (*acceptance*, *disgust*), (*joy*, *sadness*), (*anticipation*, *surprise*), and (*anger*, *fear*). They prepared the words which are given sentiment categories based

[2] http://movies.yahoo.co.jp/

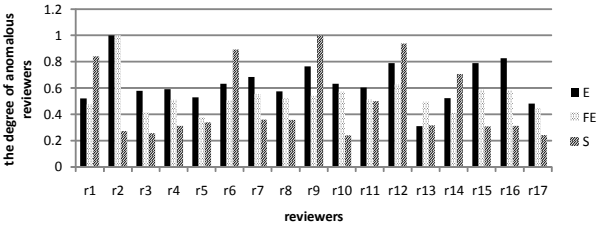


Fig. 1. The result of x in 17 reviewers

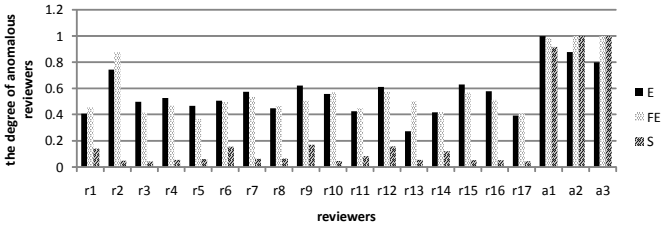


Fig. 2. The result of x in (17+3) reviewers

on this model before, and construct a sentiment dictionary studying the co-occurrence of their prepared words and wards in a huge collection. In this paper, we create a huge collection using user reviews other than reviews in the dataset, and construct a sentiment dictionary. Some words in a review are quantified, and then a review is quantified by integrating those quantified words. [2] created four axes using pairs of a sentiment category and its opposite, but we created 8-dimensional vectors using each sentient category. We normalize them in the way that the sum of the score of a sentiment category and that of its opposite sentiment category equals 1.

4.2 Result

In this experiment, we use Euclidean distance as *distance* in the three forms of evaluations. To compute feature amounts, we iterated updating just 20 times because the difference of each x_p became sufficiently small.

Detecting anomalous subjects. Fig. 1 shows all reviewers' x_p of using dataset which only includes 17 reviewers of original data and Fig. 2 shows those of using dataset which include 17 reviewers of original data and three reviewers we created. Here, we normalize the scores in the way that the max of scores equals to 1. Subjects from r1 to r17 means the reviewers in original data and those from a1 to a3 means the reviewers we created. Fig. 2 shows that all of reviewers we created have higher scores than reviewers in original data in any forms of quantified evaluations. Thus, our method can detect such anomalous reviewers using any forms of quantified evaluations. In addition, Fig. 1 and Fig. 2 show that r2 is detected as an anomalous reviewer in both

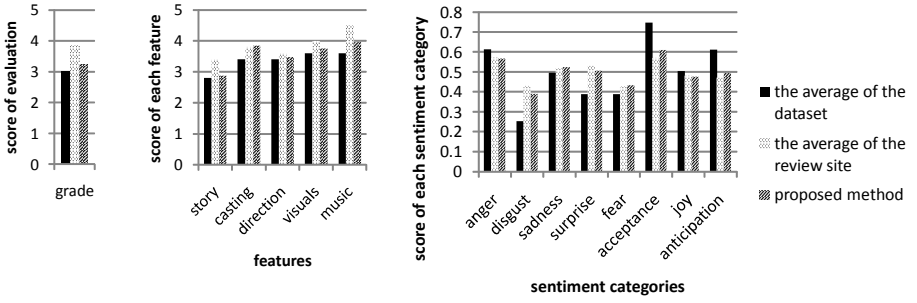


Fig. 3. The result of *Mamma Mia*! of Y in (17+3) reviewers

Table 1. Each evaluation of *Mamma Mia*!

reviewer	E	FE	S
r4	4	3, 4, 3, 4, 4	0.569, 0.411, 0.540, 0.535, 0.430, 0.588, 0.459, 0.464
r5	4	3, 5, 4, 4, 5	0.543, 0.417, 0.514, 0.516, 0.456, 0.582, 0.485, 0.483
a1	3	3, 5, 3, 5, 1	0.897, 0.214, 0.323, 0.148, 0.102, 0.785, 0.676, 0.851
a2	2	4, 1, 2, 3, 5	0.255, 0.108, 0.545, 0.671, 0.744, 0.891, 0.454, 0.328
a3	2	1, 2, 5, 2, 3	0.797, 0.108, 0.557, 0.067, 0.202, 0.891, 0.442, 0.932

method using E and FE. That is because r2 gives many movies harsh rating evaluations which is more negative than others. r2 describes similar movies and the previous movie in each movie and tends to evaluate each movie negatively. so r2 is not a malicious reviewer but an anomalous one.

Comparing Y_q with other summaries. We show Y_q and other summaries of an example title *Mamma Mia*! in Fig. 3. The left, center, and right figure show the output using E, FE, and S, respectively. We also show D_{pq} and x_p of each reviewers in this movie in Table. 1 and Table 2, respectively. In Table. 1, E, FE, and S represents (*grade*), (*story*, *casting*, *direction*, *visuals*, *music*), and (*anger*, *disgust*, *sadness*, *surprise*, *fear*, *acceptance*, *joy*, *anticipation*), respectively. In Table. 2, the value represents x_p of each reviewers evaluating the movie. Here, we use the average of the review site. In E and FE, the average of the review site equals to the average of each rating score provided in the review site. In S, we use all data including not only the dataset but also other data we crawled and create the average of evaluations among such all data. In this movie, we use 1041 reviews and create the average of the evaluations for S. In this case, anomalous reviewers are implicit and cannot be detected analyzing reviews just in this movie. For example, the evaluation of a1 seems no anomalous in the case of E, and it is also vague which reviewer is an anomalous reviewer in any forms of evaluations. However, our method use not only reviews in a target object but also reviews in other objects, so we can detect anomalous reviewers and can be Y_q close to the average of the evaluations.

Table 2. x of each reviewer evaluating *Mamma Mia*!

reviewer	E	FE	S
r4	0.678	1.650	0.044
r5	0.600	1.282	0.048
a1	1.290	3.456	0.740
a2	1.131	3.514	0.807
a3	1.031	3.513	0.808

5 Conclusion

In this paper, we propose a novel model and method for summarizing reviews and detecting anomalous reviewers. Our model and method can treat both detecting anomalous reviewers and summarizing reviews of any forms uniformly. In addition, our method is unsupervised and applicable to the situation in which there are some objects evaluated by few reviewers. In the experiment, we ascertain that our method can detect not only anomalous reviewers we created but also the anomalous reviewers in a review site. In future work, we will discuss the convergence condition of our iterative algorithm and apply our model and method to another domain.

References

1. Jindal, N., Liu, B.: Opinion spam and analysis. In: Proceedings of the International Conference on Web Search and Web Data Mining, pp. 219–230. ACM, New York (2008)
2. Kawai, Y., Kumamoto, T., Tanaka, K.: Fair news reader: Recommending news articles with different sentiments based on user preference. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part I. LNCS (LNAI), vol. 4692, pp. 612–622. Springer, Heidelberg (2007)
3. Kleinberg, J.: Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM) 46(5), 604–632 (1999)
4. Lim, E., Nguyen, V., Jindal, N., Liu, B., Lauw, H.: Detecting product review spammers using rating behaviors. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 939–948. ACM, New York (2010)
5. Liu, J., Cao, Y., Lin, C., Huang, Y., Zhou, M.: Low-quality product review detection in opinion summarization. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 334–342 (2007)
6. Plutchik, R.: The nature of emotions. American Scientist 89(4), 344–350 (2001)
7. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: Fifth IEEE International Conference on Data Mining, p. 8. IEEE, Los Alamitos (2005)
8. Wang, X., Davidson, I.: Discovering contexts and contextual outliers using random walks in graphs. In: Ninth IEEE International Conference on Data Mining, ICDM 2009, pp. 1034–1039. IEEE, Los Alamitos (2009)

Genetic Algorithm for Finding Cluster Hierarchies

Christian Böhm, Annahita Oswald, Christian Richter,
Bianca Wackersreuther, and Peter Wackersreuther

Ludwig-Maximilians-University,
Department for Informatics
Oettingenstr. 67
80538 Munich, Germany
{boehm,oswald,wackersb,wackersr}@dbs.ifi.lmu.de
richterch@cip.ifi.lmu.de

Abstract. Hierarchical clustering algorithms have been studied extensively in the last years. However, existing approaches for hierarchical clustering suffer from several drawbacks. The representation of the results is often hard to interpret even for large datasets. Many approaches are not robust to noise objects or overcome these limitation only by difficult parameter settings. As many approaches heavily depend on their initialization, the resulting hierarchical clustering get stuck in a local optimum. In this paper, we propose the novel genetic-based hierarchical clustering algorithm **GACH** (**G**enetic **A**lgorithm for finding **C**luster **H**ierarchies) that solves those problems by a beneficial combination of genetic algorithms, information theory and model-based clustering. GACH is capable to find the correct number of model parameters using the Minimum Description Length (MDL) principle and does not depend on the initialization by the use of a population-based stochastic search which ensures a thorough exploration of the search space. Moreover, outliers are handled as they are assigned to appropriate inner nodes of the hierarchy or even to the root. An extensive evaluation of GACH on synthetic as well as on real data demonstrates the superiority of our algorithm over several existing approaches.

1 Introduction

A genetic algorithm (GA) is a stochastic optimization technique based on the mechanism of natural selection and genetics, originally proposed by [15]. The general idea behind a GA is that the candidate solutions to an optimization problem (called *individuals*) are often encoded as binary strings (called *chromosomes*). A collection of these chromosomes forms a *population*. The evolution initially starts from a random population that represents different individuals in the search space. In each generation, the fitness of every individual is evaluated, and multiple individuals are then selected from the current population based on Darwin's principle "Surviving of the fittest". These individuals build the mating pool for the next generation. The new population is then formed by the application of recombination operations like *crossover* and *mutation*. A GA commonly terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. An excellent survey of GAs along with the programming structure used can be found in [10].

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 349–363, 2011.

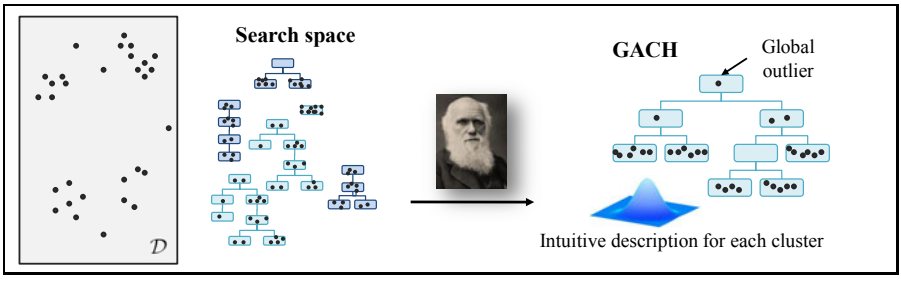


Fig. 1. The search space of the solutions for the hierarchical clustering problem is extremely large. Genetic algorithms help to determine the global optimum. GACH is a genetic algorithm for finding the most meaningful cluster hierarchy. Outliers are assigned to appropriate inner nodes. Each cluster content is described by an intuitive description in form of a PDF.

GAs have been successfully applied to a variety of challenging optimization problems, like image processing, neural networks and machine learning, etc. [21,26,2]. Solving the NP-hard clustering problem makes GA therefore a natural choice such as in [17,24,6,18,23]. The clustering research community focuses on clustering methods where the grouped objects are described by an intuitive model of the data [7] or clustering methods that are particularly insensitive to outliers [9]. Moreover, several approaches have also addressed the question, how to avoid difficult parameter settings such as the number of clusters, e.g. [22,14,3,4]. Most of them meet this question by relating the clustering problem with the idea of data compression.

Here we present a novel genetic algorithm for finding cluster hierarchies, called GACH. We use an information-theoretic fitness function to effectively cluster data into meaningful hierarchical structures without requiring the number of clusters as an input parameter. Our major contributions are:

- *Fitness*: The fitness of different chromosomes is optimized using an optimization technique that is based on the Minimum Description Length (MDL) principle.
- *No difficult parameter-setting*: Besides the parameters that are specific for a GA, GACH requires no expertise about the data (e.g. the number of clusters).
- *Flexibility*: By the use of a GA-based stochastic search GACH thoroughly explores the search space and is therefore flexible enough to find the correct hierarchical cluster structure and is insensitive to the initialization.
- *Outlier-robust*: Outliers are assigned to the root of the cluster hierarchy or to an appropriate inner node, depending on the degree of outlierness.
- *Model description*: The content of each cluster is described by a PDF.

The rest of the paper is structured as follows: Section 2 reviews some well-known approaches in the field of hierarchical and information-theoretic clustering including several genetic algorithms. In Section 3 we face the general idea behind genetic algorithms and present our proposed method GACH which searches for optimal hierarchical clustering results regarding accuracy. An extensive evaluation of GACH including method comparison and benchmarking of synthetic as well as real data is provided in Section 4 followed by the conclusion of the paper in Section 5.

2 Related Work

Although a huge amount of clustering approaches are available today, almost all of them suffer from at least one of the following drawbacks: they are restricted to partitioning clustering, and/or they are sensitive to outliers, and/or they need user-defined parameters (e.g. the number of clusters). In this section, we survey the work on hierarchical and model-based clustering and summarize important beneficial results from information theory in the field of clustering. Finally, we focus on GAs that were designed for solving the optimization problem of finding a meaningful clustering.

Hierarchical Clustering. A widespread approach to hierarchical clustering is Single Link [16]. It produces a graphical output, the so-called *dendrogram*. Cuts through the dendrogram at various levels obtain partitioning clusterings. However, for complex datasets it is hard to define appropriate splitting levels, which correspond to meaningful clusterings. Furthermore, outliers may cause the well-known Single Link effect. Also, for large datasets, the fine scale visualization is not appropriate. OPTICS [1] avoids the Single Link effect by requiring a minimum object density for clustering, i.e. $MinPts$ number of objects are within a hyper-sphere with radius ϵ. Additionally, it provides a more suitable visualization, the so-called *reachability plot*. However, the right choice of the parameters is not intuitive and has significant impact on the performance of the algorithm and the accuracy of the results. Furthermore, the problem that only certain cuts represent useful clusterings still remains unsolved.

Model-based Clustering. Model-based clustering assumes that the data is generated by a finite mixture of underlying probability distributions such as multivariate normal distributions. A commonly used algorithm for model-based clustering is the Expectation-Maximization (EM) algorithm [7]. After a suitable initialization, EM iteratively optimizes a mixture model of k Gaussians until no further significant improvement of the log-likelihood of the data can be achieved. Two common problems of EM are (1) the algorithm may get stuck in a local optimum and (2) the quality of the result strongly depends on an appropriate choice of k. Besides the classical EM, multiple hierarchical extension can be found in the literature [25,5,11]. However, each of these approaches needs a suitable parameter setting for the number of hierarchy levels.

Information-theoretic Clustering. Difficult parameter settings are often avoided by information-theoretic clustering. X-Means [22], G-Means [14] and RIC [3] try to find the optimal k in partitioning clustering by balancing data likelihood and model complexity. This sensitive trade-off is rated by model selection criteria, e.g. Minimum Description Length (MDL) [12]. The RIC algorithm uses MDL to allow for defining a coding scheme for outliers and to identify non-Gaussian clusters. However, these methods are not hierarchical but only partitioning methods. An EM-like algorithm for information-theoretic hierarchical clustering, called ITCH, is presented in [4]. After intitalization ITCH rearranges the hierarchy in a Greedy-like search which often converges only to a local optimum.

Genetic Clustering Algorithms. A genetic k-means algorithm was introduced by Krishna and Murty [17]. Scheunders [24] published a genetic variant of the c-means clustering algorithm. Some kind of semi-supervised genetic clustering was presented by [6],

which is also a k-means based approach. In [18] the authors try to improve a fitness function concerning the space restrictions on the one hand and the building blocks on the other hand. One recent approach is the one by Pernkopf and Bouchaffra [23], which combines the benefits of a GA with model-based clustering to find a nearly optimal solution for a given number of clusters. With the help of a MDL criterion the correct number of clusters is determined fully automatically. All these methods are only applicable to partitioning clustering or suffer from the problem that human interaction is still necessary to enter a suitable k for the number of clusters resulting from a fixed length of chromosomes. The detection of noise and outliers is not supported at all.

3 GACH – Genetic Algorithm for Finding Cluster Hierarchies

In this section, we describe the basic components of a GA and introduce necessary modifications to use a GA on cluster hierarchies. Finally, we present GACH as an algorithmic combination of all this components.

3.1 Chromosomal Representation of Cluster Hierarchies

Each chromosome specifies one solution to a defined problem. For GACH, a chromosome is the encoding of a hierarchical cluster structure (HCS), that has to address the three following features:

- Storage of k clusters, where k is an arbitrary number of clusters.
- Representation of the hierarchical relationship between clusters forming a tree $\mathcal{T}$ of clusters.
- Encoding of the cluster representatives, i.e. the parameters of the underlying PDF. For GACH, we represent each cluster by a Gaussian PDF. Note that our model can be extended to a variety of other PDFs, e.g. uniform or Laplacian.

With these requirements a chromosomal representation of a HCS is defined as follows:

Definition 1 (Chromosomal HCS)
*(1) A **chromosomal HCS** HCS_{Chrom} is a dynamic list storing k cluster objects.*
(2) Each cluster C holds references to its parent cluster and to its subclusters. Besides that, the level l_C for a cluster C denotes the height of the descendant subtree. This implies that the root has the highest level and the leaves have level 0.
(3) The parameters of the underlying Gaussian PDF of cluster C, the mean value μ_C and σ_C, are modeled as additional parameters of the cluster object C.
(4) Each cluster C is associated with a weight W_C, where $\sum_{i=0}^{k-1} W_{C_i} = 1$.

The underlying PDF of a cluster C is a multivariate Gaussian in a d-dimensional data space which is defined by the parameters μ_C and σ_C (where μ_C and σ_C are vectors from a d-dimensional space) by the following formula:

$$N(\mu_C, \sigma_C, x) = \prod_{1 \le i \le d} \frac{1}{\sqrt{2\pi\sigma_{C,i}^2}} \cdot \mathbf{e}^{-\frac{(x_i - \mu_{C,i})^2}{2\sigma_{C,i}^2}}$$

GACH assigns each point x in a dataset $\mathcal{D}$ *directly* to that cluster $C \in HCS_{Chrom}$ the probability density of which is maximal at the position of x:

$$C(x) = \arg \max_{C \in HCS_{Chrom}} \{W_C \cdot N(\mu_C, \sigma_C, x)\}.$$

3.2 Initialization of GACH

Basically the initial set of a population consists of a randomly generated set of individuals. This strategy is also processed by GACH, where in a first step a random number $\tilde{k}$ of clusters is selected for each structure HCS_{Chrom}. Then a simple k-means algorithm divides the dataset into $\tilde{k}$ clusters that act as the leafs of the initial hierarchy. Finally, these clusters are combined by one additional root cluster. Hence, the initialization process results in a 2-level hierarchy that consists of $\tilde{k} + 1$ nodes. Each cluster C is described by random parameters and is associated to a weight $W_C = \frac{1}{k}$.

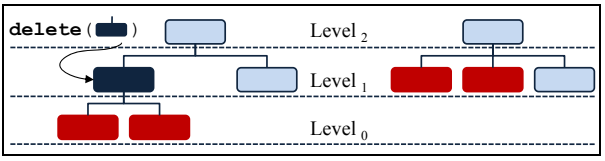
3.3 Reproduction

In order to generate the next population of cluster hierarchies GACH uses several genetic operators: mutations (`delete`, `add`, `demote` and `promote`) and `crossover`. We define these operators particularly for the hierarchical clustering problem here.

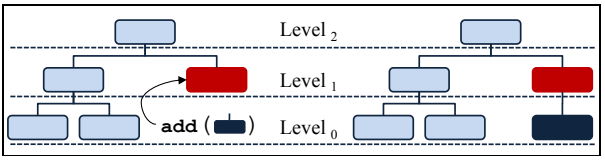
The `delete` operator deletes a specific cluster C (except the root) with a deletion rate p_{del} from the HCS. This results in structure HCS′ that does not contain the cluster C any more. This proceeding is illustrated in Figure 2(a). Here, the cluster C is marked by dark blue color. The level of each direct and undirect subcluster of C (marked in red) is decreased by 1. The former parent node of C, the root node in our example, becomes the parent node of all direct subclusters of C.

The operator `add` adds direct subclusters to an arbitrary cluster C of the hierarchy with an add rate p_{add} (normally $p_{del} = p_{add}$). The number of added subclusters is bounded by an upper limit value max_{new}. Figure 2(b) illustrates an example for the application of the `add` operator to a HCS. One subcluster (marked in dark blue color) is added to the red cluster. Since the cluster content of a added subcluster C_{add} is covered by the cluster content of C, we calculate random parameters based on μ_C and σ_C. In particular, we add a random factor r to both parameters, where r is a vector from a d-dimensional space: $\mu_{C_{add}} = \mu_C + r \qquad \sigma_{C_{add}} = \sigma_C + r$.

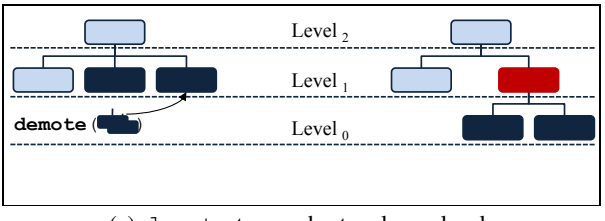
The third mutation operator is the `demote` operator, that can be motivated as follows: Assume a dataset consisting of three clusters C_1, C_2 and C_3, where C_1 holds a large number of objects, clusters C_2 and C_3 are smaller ones but they are locally close to each other. Then one could create a HCS with one root node and C_1, C_2 and C_3 as direct subclusters (cf. Figure 2(c)) which provides only a very coarse view of the dataset. But, if we combine the two smaller clusters (marked in dark blue) and demote them with a demote rate p_{dem} to a lower level with a common parent cluster (marked in dark red), we are able to get a more detailed look on our data. The parameters of the inserted cluster are obtained by the average of the parameters of the combined clusters. Note that demoting only one cluster is equal with the `add` operator. Hence, we apply `demote` on at least two clusters.



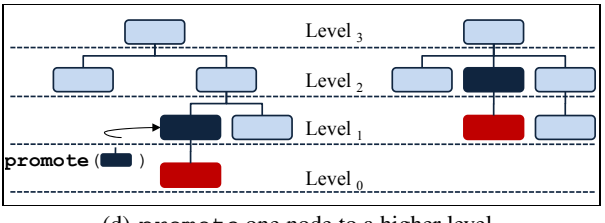
(a) `delete` one node of the hierarchy.



(b) `add` one node to the hierarchy.



(c) `demote` two nodes to a lower level.



(d) `promote` one node to a higher level.

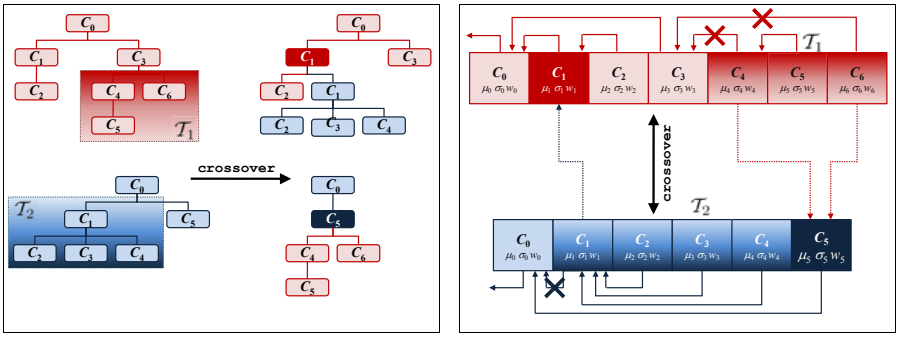
Fig. 2. Summarization of the mutation operators used for GACH

The `promote` operator lifts a cluster C from a lower level to the level right above with a promotion rate p_{pro}, if and only if C is at least two levels underneath the root cluster. Consequently all subclusters of C are lifted accordingly. In Figure 2(d) the dark blue cluster is promoted from level 1 to level 2. Hence also the red subcluster is lifted one level above. The parent of the parent node of the dark blue cluster (here the root node) becomes the parent node of C in the resulting hierarchy HCS′, together with the correct rearrangement of all subclusters.

The operator `crossover` exchanges information among two different structures. In general the information of two different chromosomes is combined in order to obtain a new individual with superior quality. GACH performs a crossover between two selected hierarchies HCS_1 and HCS_2 with a crossover rate p_{co} as follows:

1. Remove a selected subtree $\mathcal{T}_1$ entirely from HCS_1.
2. Remove a selected subtree $\mathcal{T}_2$ entirely from HCS_2.
3. Select a random node in HCS_1 and insert $\mathcal{T}_2$.
4. Select a random node in HCS_2 and insert $\mathcal{T}_1$.

Figure 3(a) illustrates this procedure exemplarily for two selected hierarchies. The subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ are removed from the red and the blue HCS respectively. $\mathcal{T}_1$ is then inserted into the blue HCS as subtree of the dark blue node. Analogously $\mathcal{T}_2$ is inserted as subtree of the dark red cluster in the red HCS. Figure 3(b) describes the same procedure w.r.t. a chromosomal representation of both hierarchies. For simplicity, only the pointers to the parent cluster are considered.



(a) Hierarchical representation. (b) Chromosomal representation.

Fig. 3. The `crossover` operator for two selected hierarchies. The subtree $\mathcal{T}_1$ of the red hierarchy is exchanged with the subtree $\mathcal{T}_2$ of the blue hierarchy, visualized by a hierarchical (3(a)) and a chromosomal representation (3(b)).

3.4 Fitness Function

Following the Darwin's principle "Survival of the fittest" naturally only individuals with highest fitness can survive and those that are weaker become extinct. A GA adopts this aspect of evolution by the use of a fitness function. GACH uses the $hMDL$ criterion formalized in [4] which evaluates the fitness of a chromosomal HCS by relating the clustering problem to that of data compression by Huffman Coding. The authors define the coding scheme for a cluster hierarchy as follows:

$$hMDL_{HCS} = \sum_{C \in HCS} \left(cost(C) - nW_C \log_2(W_C) - \log_2(\sum_{x \sqsubseteq \text{parent of } C} W_x) \right)$$

The coding cost for each cluster $C \in HCS$ is determined separately and summed up to the overall coding cost of the complete HCS. Points that are directly assigned to the cluster C together with the parameters μ_C and σ_C of the underlying Gausssian PDF are coded by $cost(C)$. The point to cluster assignment is coded by the so-called ID cost of each data point $x \in C$ and are given by $-nW_C \log_2(W_C)$ where W_C is the weight of cluster C and n the number of points. The binary logarithm is used to represent the code length in bits. Clusters with higher weight are coded by a short code pattern whereas longer code patterns are assigned for smaller clusters with lower weight. The ID cost for the parameters are formalized by $-\log_2(\sum_{x \sqsubseteq \text{parent of } C} W_x)$ whereas constant ID cost are defined for the parameters of the root node.

The better the statistical model (the HCS) fits to the data the higher the compression rate, and thus the lower the coding cost are. Using this coding scheme as fitness function ensures the selection of that chromosome HCS_{Chrom} that fits best to the data.

3.5 Selection

The selection function chooses the individuals to form the offspring population out of a set of given individuals, according to their fitness. For GACH, we use the well-known weighted roulette wheel strategy [19]. Imagine that each HCS_{Chrom} represents a number on a roulette wheel, where the amount of numbers refers to the size of the population. In addition, we assign a weight to each number on the roulette wheel, depending on the fitness of the underlying chromosome. That means that the better the fitness of a chromosome, the higher its weight on the roulette wheel, i.e. the higher the chance to get selected for the offspring population. Note that there is the chance that one chromosome is selected multiple times. GACH forms a new population that has as much individuals as the former population.

3.6 Algorithmic Description

Now we are putting the pieces together to define the algorithmic procedure of GACH, summarized in Algorithm 1. An initial population is built as described in Section 3.2. This population is evaluated according to the fitness function introduced in Section 3.4 which means that GACH determines the coding cost for each cluster hierarchy of the population. In order to optimize the point to cluster assignment of each HCS and to provide an additional model of the data, we apply an hierarchical EM algorithm on each cluster structure, as suggested in [4].

Algorithm 1. GACH

1: $count_{pop} \leftarrow 0$
2: initialize $population(count_{pop})$
3: evaluate $population(count_{pop})$
4: **while** $(count_{pop} \leq pop_{max})$ **do**
5: $\quad count_{pop} \leftarrow count_{pop} + 1$
6: $\quad$ select $population(count_{pop})$ from $population(count_{pop} - 1)$
7: $\quad$ reproduce $population(count_{pop})$
8: $\quad$ evaluate $population(count_{pop})$
9: **end while**

The population resulting from the initialization undergoes several mutation and crossover operations within pop_{max} number of generations in an iterative way. In each iteration the next population is selected according to the weighted roulette wheel strategy (cf. Section 3.5) and undergoes several reproduction procedures (cf. Section 3.3). Each operation is processed with a certain probability which is extensively evaluated in Section 4. After optimizing the point to cluster assignment, GACH determines the

fitness of each HCS_{Chrom}. The algorithm terminates if a specified maximum number of new populations pop_{max} is reached. The experiments show that the HCS can be optimized even with small generation sizes.

4 Experimental Evaluation

Now we demonstrate that the genetic parameters (mutation rate, crossover rate and population size) do not affect the effectiveness of GACH in a major way. Nevertheless, we provide a suitable parametrization that enables the user to receive good results independent of the used dataset. Based on this, we compared the performance of GACH to several representatives of various clustering paradigms on synthetic and real world data. We selected the most widespread approach to hierarchical clustering Single Link (SL) [16], the more outlier-robust hierarchical clustering algorithm OPTICS [1] (requiring $MinPts$ and ϵ), with optimal parameters w.r.t. accuracy. Furthermore, we chose RIC [3], an outlier-robust and information-theoretic clusterer, and finally ITCH [4] which is a recent EM-based hierarchical information-theoretic clustering approach, that suffers from the problem that the result often only represents a local optimum. As ITCH strongly depends on its initialization, we used the best out of ten runs in this case. For the SL experiments, we used the Matlab implementation. OPTICS was provided by WEKA [13]. For RIC and ITCH we used the original Java implementations by the authors.

4.1 Evaluation of Genetic Parameters

We applied GACH on two different datasets to evaluate the mutation and crossover rates and the impact of the population size on the quality of the results w.r.t. the fitness function, introduced in Section 3.4. One dataset consists of 1,360 (2d)-data points that form a true hierarchy of six clusters. The second dataset covers 850 (2d)-data points that are grouped in two flat clusters. For each experiment, we present the mean $hMDL$ value and the corresponding standard deviation over ten runs. GACH turned out to be very robust and determines very good clustering results ($Prec > 90\%$, $Rec > 90\%$) indepent of the parametrizations.

Different Mutation Rates. We evaluated different mutation rates ranging from 0.01 to 0.05 on two different population sizes and a fixed crossover rate of 0.15. As a mutation is performed by one of the four operations `delete`, `add`, `demote` or `promote` the mutation rate is the sum of p_{del}, p_{add}, p_{dem} and p_{pro} (cf. Section 3.3). As `demote` and `promote` turned out to be essential for the quality of the clustering results p_{dem} and p_{pro} are typically parametrized by a multiple of p_{del} or p_{add}. This is due to the fact that the optimal number of clusters which is influenced by p_{del} and p_{add} is determined very fast by the fitness function, but p_{dem} or p_{pro} have an impact on the hierarchical structure of the clusters that has to be adjusted during the run of GACH. Figures 4(a) and 4(b) demonstrate that the mutation rate has no outstanding effect on the clustering result, neither on a hierarchical nor on a flat dataset. Higher mutation rates result in higher runtimes (3,388 ms for mutation rate = 0.05 vs. 1,641 ms for mutation rate = 0.01

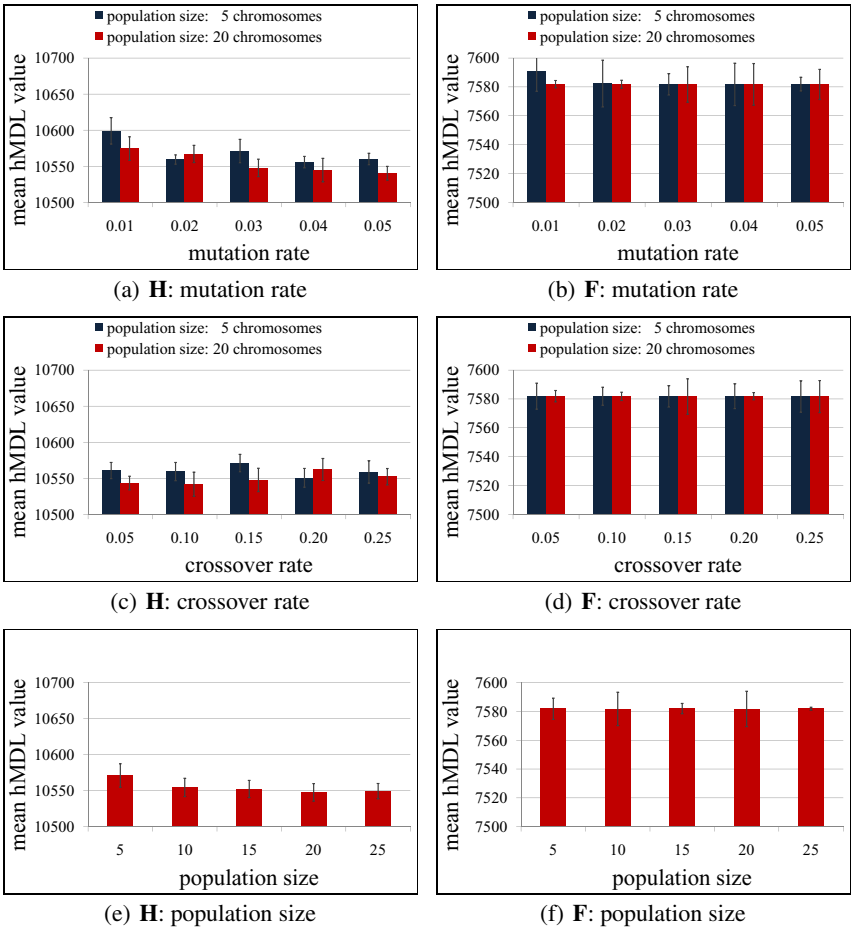


Fig. 4. Mean fitness of resulting clusterings on (**H**)ierarchical and (**F**)lat datasets w.r.t. the genetic parameters mutation rate, crossover rate and population size

on hierarchical dataset, population size = 5). However, a higher mutation rate provides more flexibility. Hence, we achieved slightly better results with a mutation rate of 0.05 ($hMDL = 10,520$) compared to a mutation rate of 0.01 ($hMDL = 10,542$).

Different Crossover Rates. We compared different crossover rates p_{co} ranging from 0.05 to 0.25 in combination with a mutation rate of 0.05 on two different population sizes. Figures 4(c) and 4(d) show that the performance of GACH is almost stable w.r.t. the different parameterizations of p_{co}. Especially on a flat dataset, a higher p_{co} has no impact on the clustering result. GACH achieved a nearly optimal $hMDL$ value in almost every run, even for relatively small population sizes. Higher p_{co} values enable GACH to examine the search space more effectively as the crossover between two strong individuals produces an even fitter individual. Therefore, we need less generations to find good clustering results, e.g. the result of GACH on the hierarchical dataset

using five structures was determined after 75 generations (1,993 ms per generation) with $p_{co} = 0.05$, and after 61 generations (2,553 ms per generation) with $p_{co} = 0.25$.

Different Population Sizes. We tested the impact of the population size on the quality of the clustering result. We used populations that cover 5, 10, 15, 20 and 25 hierarchical cluster structures in combination with a mutation rate of 0.05 and a crossover rate of 0.15. Figures 4(e) and 4(f) show again the mean $hMDL$ value over ten runs for each population size on two different datasets. Especially the plot of the hierarchical dataset demonstrates that a higher population size tends to produce better results, which can be explained by the fact that a higher population size provides more variation opportunities whereby a global optimum can be reached easier. However, a large number of chromosomes cause a considerable amount of runtime. One generation covering five chromosomes took 2,462 ms on average, the computation of a generation consisting of 25 chromosomes took 9,229 ms. Hence, we use a population size consisting of ten cluster structures in combination with a mutation rate of 0.05 and a crossover rate of 0.15 in the following experiments.

4.2 Competitive Performance of GACH

For these experiments we use two different synthetic datasets DS_1 and DS_2. DS_1 is composed of 987 (2d)-data points that form a hierarchy of six clusters surrounded by local and global noise (cf. Figure 5(a)). DS_2 consists of 1,950 (2d)-data points that are grouped in three flat strongly overlapping clusters (cf. Figure 5(b)). For each dataset, the clustering results were assessed against a ground-truth classification. To compare the clustering results produced by different approaches, we chose measures that comprise fix bounds. More precisely, we selected the following recently proposed information-theoretic methods [20]: the Normalized Mutual Information (NMI), the Adjusted Mutual Information (AMI), which corrects the NMI for randomness. Both measures have value 1 if the resulting clustering corresponds exactly to the ground-truth classification, and 0 when both clusterings are totally independent. Furthermore, we chose the well-known Precision (Prec) and Recall (Rec) from information retrieval. Finally, we use the DOM [8] value that corresponds to the number of bits required to encode the class labels when the cluster labels are known, i.e. low DOM values represent good clustering results.

Evaluation w.r.t. Dataset $\mathbf{DS_1}$. These experiments demonstrate that GACH performs at least as good as the parameter dependent approach OPTICS ($MinPts = 6, \epsilon = 0.9$) and the recently proposed method ITCH, while being much more accurate than RIC and SL. The reachability plot of OPTICS is provided in Figure 5(c), the SL dendrogram is shown in Figure 5(e). Although DS_1 seems to be an easy to cluster dataset, SL and RIC fail in assigning the local and global outliers to the correct clusters. Both, GACH and ITCH were able to determine the right cluster structure (shown in Figure 5(g)). However, GACH outperformed ITCH w.r.t. accuracy, as GACH results in a different points to clusters assignment. Therefore, GACH shows the best performance on DS_1 concerning the information-theoretic measures NMI and AMI. Moreover, the values for Precision and Recall indicate that 94% of all data points are assigned to the true cluster

and 95% of the cluster contents were detected correctly by GACH. Finally, this result can be coded most efficiently as stated by the low DOM value of 0.4193. The evaluation on DS_1 is summarized in Table 1.

Table 1. Performance of GACH on DS_1

	GACH	ITCH	RIC	OPTICS	SL
NMI	**0.9346**	0.9265	0.8673	0.9045	0.9110
AMI	**0.9159**	0.8999	0.7678	0.8662	0.8997
PREC	0.9404	0.9222	0.6438	0.8626	**0.9555**
REC	**0.9514**	0.9422	0.7720	0.9200	0.9169
DOM	**0.4193**	0.4454	0.6638	0.4960	0.4765

Evaluation w.r.t. Dataset $\mathbf{DS_2}$. Neither OPTICS nor SL were able to detect the true cluster structure of DS_2. Both fail because of a massive Single Link effect and therefore the reachability plot provided by OPTICS (cf. Figure 5(d)) and the dendrogram produced by SL (cf. Figure 5(f)) do not uncover any cluster structure which leads to a clustering where all objects belong to the same cluster. Also RIC determines only one single cluster. ITCH separates the dataset into only two clusters. In contrast, GACH identifies all clusters in the dataset correctly. Hence, GACH turned out to be the only algorithm that handles datasets with strongly overlapping clusters successfully. It shows the best values w.r.t. information-theoretic criterions, while being very accurate. Its result causes only a coding cost of 0.3325 compared to more than 0.9 for almost all other approaches. The evaluation on DS_2 is summarized in Table 2.

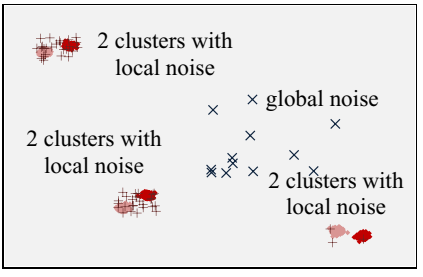
Table 2. Performance of GACH on DS_2

	GACH	ITCH	RIC	OPTICS	SL
NMI	**0.6698**	0.6316	0.0000	0.0000	0.0000
AMI	**0.5877**	0.4030	0.0000	0.0000	0.0000
PREC	**0.9184**	0.8227	0.2130	0.5016	0.6750
REC	**0.9226**	0.8913	0.4615	0.4615	0.4615
DOM	**0.3325**	0.4226	0.9184	0.9184	0.9184

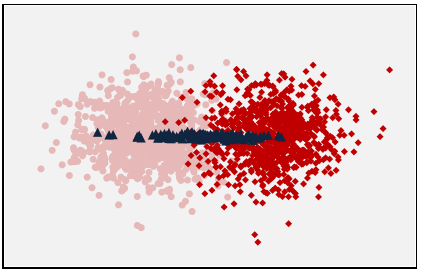
4.3 Application of GACH on Real World Data

We tested the practical application of GACH on several real world datasets. Due to space restrictions, we selected the high dimensional *Wine* dataset[1] for presentation. It contains 178 (13-d)-data objects resulting from a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. Hence, a ground-truth classification structures the data into one root node covering the whole dataset and three subclusters defining the three cultivars. This structure was only determined by GACH resulting in high validity values (cf. Table 3). Most of the competitors did not even find

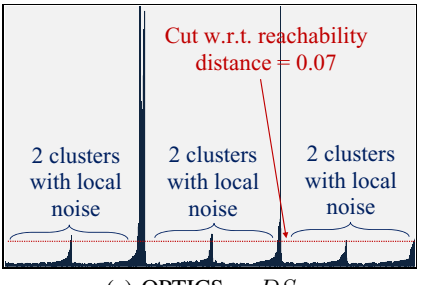
[1] http://archive.ics.uci.edu/ml/datasets/Wine



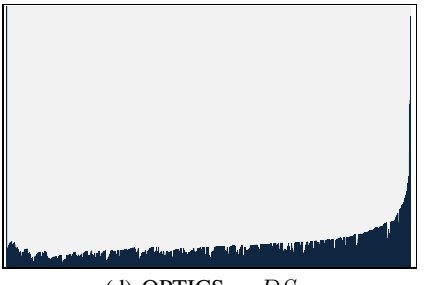
(a) Plotted DS_1.



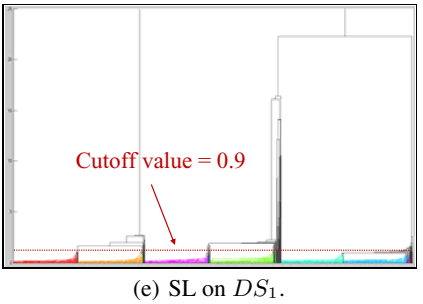
(b) Plotted DS_2.



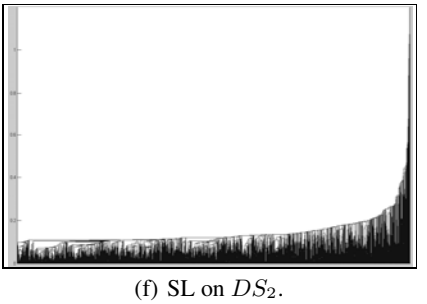
(c) OPTICS on DS_1.



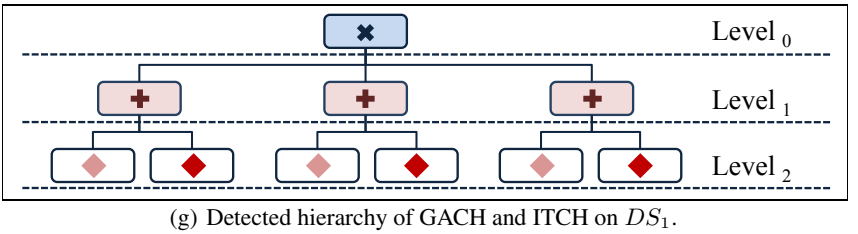
(d) OPTICS on DS_2.



(e) SL on DS_1.

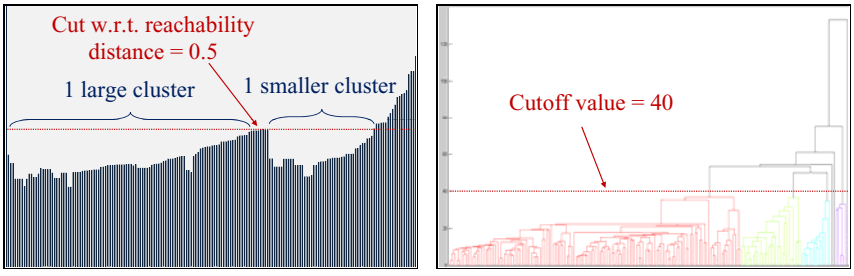


(f) SL on DS_2.



(g) Detected hierarchy of GACH and ITCH on DS_1.

Fig. 5. Competitive evaluation of GACH on two different synthetic datasets. DS_1 forms a hierarchy including local and global noise, DS_2 is a flat dataset of three overlapping clusters.



(a) OPTICS on *Wine*. (b) SL on *Wine*.

Fig. 6. Competitive evaluation of GACH on real-world data

the right number of clusters. For example, OPTICS uncovers two different clusters (cf. Figure 6(a)) and RIC even merges all data points in only one single cluster. SL detects four different clusters concerning a cutoff value of 40. Besides the *Wine* dataset, GACH turned out to be applicable in many application domains.

Table 3. Performance of GACH on *Wine*

	GACH	ITCH	RIC	OPTICS	SL
NMI	**0.7886**	0.7615	0.0000	0.5079	0.3852
AMI	**0.7813**	0.6912	0.0000	0.4817	0.3485
PREC	0.9401	**0.9737**	0.1591	0.7466	0.5309
REC	**0.9326**	0.8596	0.3989	0.6966	0.5337
DOM	0.3631	0.3285	**1.1405**	0.6853	1.0740

5 Conclusion

We proposed GACH – a genetic algorithm for finding cluster hierarchies, that combines the benefits of genetic algorithms, information theory and model-based clustering being an efficient and accurate hierarchical clustering technique. As GACH uses a MDL-based fitness function, it can be easily applied to real world applications without requiring any expertise about the data, like e.g. the real number of clusters. By the integration of an EM-like strategy, the content of all clusters is described by an intuitive model. GACH handles outliers by assigning them to appropriate inner nodes of the hierarchy, depending on their degree of outlierness. Our experimental evaluation demonstrates that GACH outperforms a multitude of other clustering approaches.

References

1. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: OPTICS: Ordering Points To Identify the Clustering Structure. In: SIGMOD Conference, pp. 49–60 (1999)
2. In: Bäck, T. (ed.) Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, USA, July 19-23. Morgan Kaufmann, San Francisco (1997)

3. Böhm, C., Faloutsos, C., Pan, J.Y., Plant, C.: Robust Information-theoretic Clustering. In: KDD, pp. 65–75 (2006)
4. Böhm, C., Fiedler, F., Oswald, A., Plant, C., Wackersreuther, B., Wackersreuther, P.: ITCH: Information-Theoretic Cluster Hierarchies. In: ECML/PKDD, vol. (1), pp. 151–167 (2010)
5. Chardin, A., Pérez, P.: Unsupervised Image Classification with a Hierarchical EM Algorithm. In: ICCV, pp. 969–974 (1999)
6. Demiriz, A., Bennett, K.P., Embrechts, M.J.: Semi-supervised clustering using genetic algorithms. In: Artificial Neural Networks in Engineering, pp. 809–814 (1999)
7. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological) 39(1), 1–38 (1977)
8. Dom, B.: An Information-Theoretic External Cluster-Validity Measure. In: UAI, pp. 137–145 (2002)
9. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: KDD, pp. 226–231 (1996)
10. Filho, J.R., Alippi, C., Treleaven, P.: Genetic Algorithm Programming Environments. IEEE Computer 27, 28–43 (1994)
11. Goldberger, J., Roweis, S.T.: Hierarchical Clustering of a Mixture Model. In: NIPS (2004)
12. Grünwald, P.: A tutorial introduction to the minimum description length principle. CoRR math.ST/0406077 (2004)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations 11(1), 10–18 (2009)
14. Hamerly, G., Elkan, C.: Learning the k in k-means. In: NIPS (2003)
15. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1992)
16. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
17. Krishna, K., Murty, M.N.: Genetic K-means algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B 29(3), 433–439 (1999)
18. Lorena, L.A.N., Furtado, J.C.: Constructive Genetic Algorithm for Clustering Problems. Evolutionary Computation 9(3), 309–328 (2001)
19. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, Heidelberg (1996)
20. Nguyen, X.V., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: ICML, p. 135 (2009)
21. Pal, S.K., Bhandari, D., Kundu, M.K.: Genetic algorithms for optimal image enhancement. Pattern Recogn. Lett. 15(3), 261–271 (1994)
22. Pelleg, D., Moore, A.W.: X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In: ICML, pp. 727–734 (2000)
23. Pernkopf, F., Bouchaffra, D.: Genetic-Based EM Algorithm for Learning Gaussian Mixture Models. IEEE Trans. Pattern Anal. Mach. Intell. 27(8), 1344–1348 (2005)
24. Scheunders, P.: A genetic c-Means clustering algorithm applied to color image quantization. Pattern Recognition 30(6), 859–866 (1997)
25. Vasconcelos, N., Lippman, A.: Learning Mixture Hierarchies. In: NIPS, pp. 606–612 (1998)
26. Whitley, L.D., Starkweather, T., Bogart, C.: Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Computing 14(3), 347–361 (1990)

A File Search Method Based on Intertask Relationships Derived from Access Frequency and RMC Operations on Files

Yi Wu[1,*], Kenichi Otagiri[1,**], Yousuke Watanabe[2], and Haruo Yokota[1]

[1] Department of Computer Science, Tokyo Institute of Technology
[2] Global Scientific Information and Computing Center, Tokyo Institute of Technology
{goi,otagiri,watanabe}@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp
http://yokota-www.cs.titech.ac.jp

Abstract. The tremendous growth in the number of files stored in filesystems makes it increasingly difficult to find desired files. Traditional keyword-based search engines are incapable of retrieving files that do not include keywords. To tackle this problem, we use file-access logs to derive intertask relationships for file search. Our observations are that 1) files related to the same task are frequently used together, and 2) a set of Rename, Move, and Copy (RMC) operations tends to initiate a new task. We have implemented a system named SUGOI, which detects two types of task, FI tasks and RMC tasks, from file-access logs. An FI task corresponds to a group of files frequently accessed together. An RMC task is generated by RMC operations and then constructs a graph of intertask relationships based on the influence of RMC operations and the similarity between tasks. In utilizing detected tasks and intertask relationships, our system expands the search results of a keyword-based search engine. Experiments using actual file-access logs indicate that the proposed approach significantly improves search results.

Keywords: file-access logs, desktop search, full-text search, task mining.

1 Introduction

The explosion in the volume of information that people handle has been accompanied by daily increases in the number of files stored in filesystems. Many of these are unstructured files, such as images, diagrams, and numerical-data files, which do not contain any appropriate text that can be used for search. As these files do not include the target keywords, traditional text-based desktop keyword search engines are not useful for finding them.

A number of systems have been developed for desktop keyword search, such as Google Desktop Search[3], Microsoft's Windows Desktop Search[5], and Spotlight on Mac OS X[1]. These systems all develop indexes for high-speed search,

* The author is currently with NTT DATA Corporation.
** The author is currently with Cowbell Engineering Corporation.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 364–378, 2011.

and some use a thesaurus and meta-information such as the file name, creation time, and file type, to improve search performance. However, it is still difficult to search for files that do not include text.

Recently, some desktop search systems, such as FRIDAL[9,10] and Connections[8], have been proposed to tackle the problem of the association between files based on the co-occurrence information derived from file-access logs. These systems try to capture the relationships between files based on users' access patterns. These approaches are effective in searching for files that do not contain text. However, sometimes the searches return irrelevant files or fail to return relevant files because infrequent but important operations for files could not be found and temporary simultaneous access of files influences the results from these approaches.

In this paper, to improve the accuracy of search results based on file-access logs, we focus on a task where a user accesses multiple files to accomplish his/her goal. In other words, most files accumulated in a filesystem must be related to individual tasks. We refer to a group of files that are used to accomplish a particular goal as a "task." Thus, in this paper, the term "task" represents a logical unit of files to be processed in a computer system, such as collating some experimental data, writing a report or article, or preparing presentation slides.

Based on the concept of task, we propose a file search system named *SUGOI*—Search by Utilizing Groups Of Interrelated files in a task—which consists of two parts: a task mining component and a file search component. The task mining component extracts tasks and discovers the interrelation between tasks from file-access logs. The file search component incorporates the task mining results within the search results of a traditional desktop search engine to achieve an accurate keyword-based file search.

We observed that 1) files related to the same task are frequently used together, and 2) a set of Rename, Move and Copy (*RMC*) operations tends to initiate a new task. Therefore, the task mining component of SUGOI extracts two types of task: *FI* (Frequent Itemset) tasks and *RMC* tasks. We then propose formulas to combine FI and RMC tasks.

We also consider the graph of intertask relationships for retrieving related files. Assuming that the greater the number of identical files accessed during different tasks, the stronger is the interrelation between these tasks, we build similarity links between these tasks. In addition, we generate RMC links between tasks, assuming that users tend to RMC files for reuse in related tasks. For example, as researchers tend to use the same graph of experimental results in both their articles and their presentation slides, they may copy the file of the graph from the article folder to the presentation folder. In this case, the copy operation indicates a strong relationship between the tasks of writing the article and of preparing the presentation. To represent such intertask relationships, the weight of an RMC link is computed considering the type of the operation and its direction. We propose a formula to handle both types of links.

After the tasks and intertask relationships have been extracted, the file search component of SUGOI expands the search results of a traditional keyword-based

search engine using the mined tasks and intertask relationships, and ranks the search results. We evaluate the proposed system using actual file-access logs. The experimental results indicate that the proposed approach significantly improves recall and F-measure.

The remainder of this paper is organized as follows. Section 2 presents related research on desktop search. Section 3 gives a detailed explanation of SUGOI and Section 4 describes the experimental results and investigation. Section 5 concludes this paper and indicates future work.

2 Related Work

With the great increase in the capacity of storage devices, the volume of all types of data is steadily increasing, with most being file-based and unstructured. Users require more effective and simpler mechanisms for managing vast numbers of files, such as locating desired files easily from files scattered across different directories. Traditional content-based search tools cannot deal with files that do not include the query keywords, and supported file formats are restricted. For the purpose of enhancing full-text search, much research has been done using file-access logs. This section describes three previous studies that used file-access logs for file search.

As mentioned in Section 1, Connections [8] is a file search system that uses contextual information with the aim of enhancing full-text search results. Connections generates a relational graph of files based on traces of filesystem calls. To discover relationships, it splits access logs into multiple relation windows and identifies the input and output files in each window by considering which operation is performed on the files. Connections then creates links with weight 1 from input files to output files or increases the weight of existing links. It uses a Basic-BFS (Breadth First Search) algorithm to propagate the weights of keyword-containing files to keyword-lacking files to expand and reorder the results generated by a full-text search engine. By contrast, we utilize file-access logs to group files into tasks and identify the interrelations between tasks rather than the relationships between files.

FRIDAL [9,10] is another system for searching keyword-lacking files by using the interfile relationship. FRIDAL exploits open/close logs in order to derive the duration for which the file is used. Assuming that files used at the same time have some relevance to each other, FRIDAL calculates the interfile relationship by using the co-occurrence data between files in access logs. It finds keyword-lacking files by using a Basic-BFS-like method. By contrast, SUGOI emphasizes tasks and only uses access patterns that occur frequently. Furthermore, we take RMC operations into account in calculating the weight of semantic links between tasks.

The iMecho [2] system performs task mining similar to SUGOI, building three types of associations: Content-based Associations (CA), Explicit Activity-based Associations (EAA), and Implicit Activity-based Associations (IAA). It reranks the results of a full-text search engine, using a random walk algorithm based on

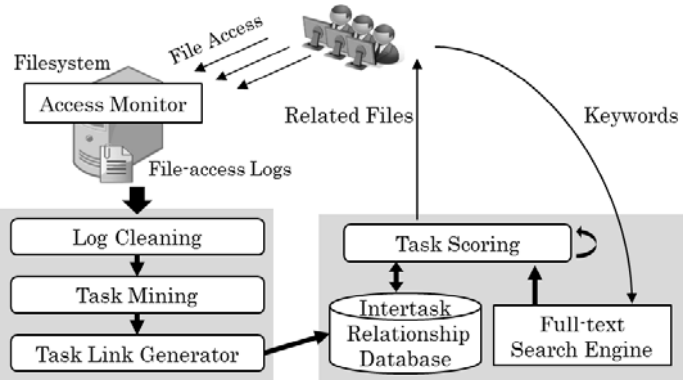


Fig. 1. Overview of SUGOI

PageRank[7]. The purpose of the task mining in iMecho is to generate IAA links between files; it does not consider the relationship between tasks. In addition, the ranking score used in iMecho is defined as the product of a full-text search result and a link analysis result, so iMecho does not extend the result of a full-text search to include non-text files. By contrast, SUGOI can extend the result of a full-text search by task mining and detecting the relevance between tasks.

3 Proposed Approach

We propose an approach for searching files by introducing the concept of "tasks." Before the search process, we cluster related files as a task and discover the correlation between tasks by exploiting file-access logs. We then expand the results from a traditional keyword-search engine using the tasks and intertask relationships. We have implemented a prototype system using the proposed approach and named it "Search by Utilizing Groups Of Interrelated files in a task" or "SUGOI" for short. An overview of SUGOI is illustrated in Fig. 1. To trace user access as file-access logs, SUGOI places an access monitor on the filesystem. The file-access logs should include access time, information to identify the client, the path of the target file, and the operation on it. After cleaning the log, the system extracts the semantic file groups as tasks, and builds weighted links between tasks. Following a user's search request, SUGOI searches for files by combining the context of tasks with the traditional full-text search results to calculate the task scores. The details of the proposed methods (apart from log cleaning) are described in the following subsections. Log cleaning is described in Section 4.1, because the process corresponds to the monitoring tool used in the experiments.

3.1 Task Mining

The purpose of task mining is to group those files that are related to the same task. We extract two types of task: FI tasks and RMC tasks. After the task

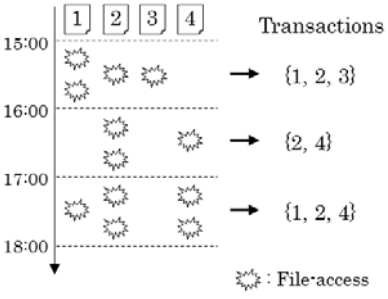


Fig. 2. Extraction of Transactions (TransactionTime = 3600 [s])

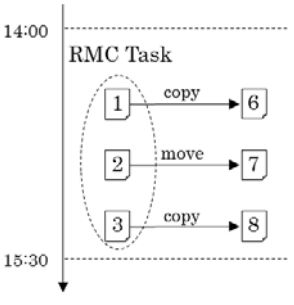


Fig. 3. Extraction of an RMC Task (RMCTaskTime = 1800 [s])

mining processes, a number of files accessed by a user belong to at least one task.

FI Task Mining. The FI task, or Frequent Itemset task, is constructed from files accessed concurrently. The point is that files related to the same task tend to be accessed frequently within short time periods of each other. For example, when a user writes an article, he/she edits a TeX file as well as EPS files containing charts and produces a PDF file. With this characteristic, we propose a method to semantically group files as FI tasks by mining the access patterns that occur frequently and simultaneously.

We first convert the file-access logs into a set of transactions (Fig. 2) by splitting the logs into several transactions with a certain duration ($TransactionTime$). To discover the frequent patterns, we apply an existing algorithm, Eclat [11], to the set of transactions. Eclat is one of the best-known algorithms for mining frequent itemsets. It finds combinations of items whose occurrence frequency is greater than some minimum support value. In this study, we determine the minimum support count ($MinSuppCnt$) to extract frequent itemsets because file-access logs last for a long period but common tasks last for a fixed period of time and the number of occurrences of each individual file should be small. Finally, we extract the maximally frequent itemsets as FI tasks.

RMC Task Mining. Assuming that the files RMCed together within a short time period are related to a task done in the past, we extract such file groups as an RMC task. To mine RMC tasks, we split access logs into multiple time windows with the same length of time and only extract the files that are the target of RMC operations (Fig. 3).

3.2 Intertask Relationship

This subsection presents our proposed formulas for weighting the similarity links and RMC links of tasks and calculating the relevance between tasks by considering the overlap and RMC operations between tasks. In addition, we consider reducing the weight of RMC links because the content of files created by RMC operations can differ from the original by modification.

Similarity Links Analysis. Assuming that tasks that are strongly interrelated use a number of common files, we weight the similarity links between tasks based on the number of duplicated files in each task. The weight of a similarity link from task i to task j, which is expressed as $sim(t_i \rightarrow t_j)$, is given by Equation (1).

$$sim(t_i \rightarrow t_j) = \frac{|t_i \cap t_j|}{|t_i|} \tag{1}$$

Here, t_i and t_j represent the file sets of task i and task j, respectively.

RMC Links Analysis. Supposing that a user RMCes files contained in related task to reuse in other tasks, RMC operations can express a strong relationship between tasks. We build RMC links of tasks by detecting an RMC operation between different tasks. To calculate the weight of RMC links, we introduce the element $rmc_f(f_m \rightarrow f_n)$, which presents the weight from file m to file n caused by an RMC operation.

$$rmc_f(f_m \rightarrow f_n) = \begin{cases} \alpha_1 \text{ if } f_m \text{ was renamed as } f_n, \\ \alpha_2 \text{ if } f_m \text{ was renamed from } f_n, \\ \beta_1 \text{ if } f_m \text{ was moved to } f_n, \\ \beta_2 \text{ if } f_m \text{ was moved from } f_n, \\ \gamma_1 \text{ if } f_m \text{ was copied to } f_n, \\ \gamma_2 \text{ if } f_m \text{ was copied from } f_n, \\ 0 \quad \text{otherwise.} \end{cases} \tag{2}$$

According to Equation (2), we assign rmc_f a constant value specified by the parameters (α_1, α_2, β_1, β_2, γ_1, γ_2). However, the degree of relevance of each file will be attenuated because the content of files coming from RMC operations probably differs from that of the original files by repeated edits.

In addition to rmc_f, we propose formulas to take into consideration the factors that cause reduced relevance. Thus, we define the reduction formulas using the elapsed time (Equation (3)), the sum of frequency of write operation handled to

file f_m and file f_n (Equation (4)), and the sum of the sizes of the changes to file f_m and file f_n after RMC operations performed (Equation (5)).

$$T(f_m, f_n) = \Delta_{time}(f_m, f_n)^{-\tau} \tag{3}$$

$$E(f_m, f_n) = \Delta_{edit}(f_m, f_n)^{-\epsilon} \tag{4}$$

$$S(f_m, f_n) = \Delta_{size}(f_m, f_n)^{-\sigma} \tag{5}$$

Here, τ, ϵ, σ are parameters. Considering the circumstances mentioned above, we calculate the weight of an RMC link by using rmc_f and the reduction functions (Equation (6)).

$$rmc(t_i \rightarrow t_j) = \sum_{(f_m, f_n) \in (t_i, t_j)} rmc_f(f_m \rightarrow f_n) * T(f_m, f_n) * E(f_m, f_n) * S(f_m, f_n) \tag{6}$$

Intertask Relevance. To calculate the degree of association between tasks, we adopt the weight of similarity links and RMC links and use the parameter $\theta (0 \leq \theta \leq 1)$ to control which element to emphasize. The proposed formula is given by Equation (7).

$$R(t_i \rightarrow t_j) = \theta * sim_t(t_i \rightarrow t_j) + (1 - \theta) * rmc_t(t_i \rightarrow t_j) \tag{7}$$

3.3 Keyword-Based File Search

In addition to files that include keywords, SUGOI can find files contained in tasks related to the keywords by combining the context and interrelation of tasks. This subsection describes the procedure for keyword-based search.

STEP 1: Identify tasks containing the keyword-containing files. The initial relevance score of a task to the given keywords is assigned using the file score given by the existing full-text search engine (Equation (8)).

$$score^0(q, t_i) = \sum_{f_m \in t_i} score_f(q, f_m) \tag{8}$$

Here, $score^0(q, t_i)$ denotes the initial score of task t_i to query q, and $score_f(q, f_m)$ denotes the relevance score of file f_m to the query. According to Equation (8), we use the summation of the file score to define the task score so that the score of the tasks that do not contain the keyword-containing files should be zero.

STEP 2: In this paper, we adopt a reflexive method based on Basic-BSF, which is used in Connections[8] and FRIDAL[9,10], to find tasks that contain related files. To find the files that do not contain keywords, we iterate the calculation of the relevance score for all tasks K times for translating the relevance ($score^0(q, t_i)$) to other tasks for emphaisizing tasks that linked with

many related tasks and giving score to tasks that do not contain keyword-containing files. At the k-th ($1 \leq k \leq K$) calculation, the relevance score of task t_i is given by Equation (9).

$$score^k(q, t_i) = score^{k-1}(q, t_i) + \sum_{t_j \in InLink(t_i)} score^{k-1}(q, t_j) * R(t_j \rightarrow t_i) \quad (9)$$

Here, $InLink(t_i)$ expresses the set of tasks whose links point to task t_i.

STEP 3: Normalize the relevance score for all t_i and output files contained in tasks that satisfy $score^K(q, t_i) > TH_{score}$ as results, where TH_{score} is a threshold parameter.

4 Experiments

4.1 Experimental Environment

To verify the efficiency of SUGOI, we conduct evaluation experiments by using actual file-access logs gathered from a shared filesystem (Windows Server 2003 SP2, NTFS) used by our research group. To monitor the file access to the server, we use a tool named FAccLog[6]. FAcclog records the logs by monitoring access to the OS and the LAN adapter. Logging can be done almost in real time and includes read, write, create, delete, and rename operations. The full path of the target file is also included, but in the case of a rename, the path is recorded as "path before rename $\gg$ path after rename". Since the raw logs have noise and some necessary information is lacking, we apply the log-cleaning process before the main mining process.

The logs created by machine access mostly generate an incorrect task mining result, deriving groups of files with little reference to each other. Such kinds of access often come from background processes such as virus scanning, extracting indexes by a desktop search engine or making backups. In addition, in many cases, a lot of file access will occur in a short period. Therefore, we use two thresholds (TH_{min} and TH_{sec}) to eliminate the background machine access from logs. If the number of accesses occurring in a one minute/second range is larger than TH_{min}/TH_{sec}, all of the log entries are ignored. We also prepare a filter for detecting machine access by file extension.

The raw logs only distinguish rename operations; they do not move and copy. To find move operations, we treat entries whose directory part of the path changed after the rename as move operations. To detect copy operations, we look for a pattern of a create log entry occurring after a read log entry with the same filename, and treat it as a copy operation.

The implementation of SUGOI uses an existing full-text search engine named HyperEstraier[4]. In addition to plain text and HTML files, we can optionally search PDF, DOC, DOCX, XLS, XLSX, PPT, and PPTX.

4.2 Experimental Setup

The purpose of the experiments was to verify the effect of utilizing file-access logs for file search. We used datasets with keywords and lists of relevant files

Table 1. Experimental Datasets

Dataset	# log entries	# files	# files available for full-text search	# relevant files
A	3591	201	137	70
B	2808	416	113	25
C	3424	318	276	32
D	5911	764	311	84
E	8203	642	335	244
F	5123	3422	1152	227
G	13102	3258	338	73

Table 2. Average F-measure (FI Task Mining)

TransactionTime [s]	900	1800	3600	5400	7200
MinSuppCnt=2	0.543	0.513	0.507	0.470	0.506
MinSuppCnt=3	0.452	0.432	0.450	0.411	0.453

provided by seven testers. The system was evaluated by comparing the relevant files except the files that were deleted from the filesystem. A summary of the experimental datasets is given in Table 1. These datasets were gathered from April to July 2010. There are more than 2 GB of raw data and the summary is of the logs after log cleaning.

The parameters below were fixed during the experiments. In log cleaning, $TH_{min} = 30$, $TH_{sec} = 5$. To calculate the weight of RMC links, $(\tau, \epsilon, \sigma) = (0, 0, 0)$. In keyword-based search, $TH_{score} = 0$, $K = 3$.

4.3 Evaluation of Task Mining

To expand the full-text search results, we detect two types of tasks and derive the intertask relationships from file-access logs. As files contained in the same task are identified with each other, the results of task mining should be important. We first set up these experiments to acquire appropriate values of parameters used for task mining. The other parameters were fixed as follows: $(\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2) = (1.0, 1.0, 1.0, 1.0, 1.0, 1.0)$, $\theta = 0.5$.

Parameter Tuning for FI Task Mining. To determine the value of $TransactionTime$ and $MinSuppCnt$, we only used FI tasks, and experiments were performed in the combinations of $TransactionTime = \{900, 1800, 3600, 5400, 7200\}$ [s] and $MinSuppCnt = \{2, 3\}$.

The averages of the F-measure are depicted in Fig. 4. The $MinSuppCnt = 2$ case performed better than the $MinSuppCnt = 3$ case, with the value of $TransactionTime$ held constant. Increasing $MinSuppCnt$ caused the number

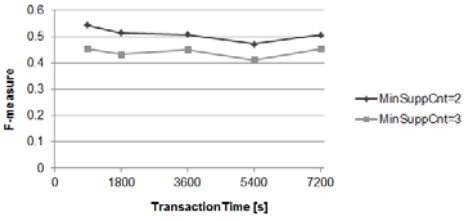


Fig. 4. Experimental Results for FI Task Mining

Table 3. Average F-measure (RMC Task Mining)

RMCTaskTime [s]	60	180	300	600	1800
Precision	0.776	0.758	0.762	0.762	0.756
Recall	0.669	0.674	0.673	0.675	0.684
F-measure	0.684	0.681	0.679	0.684	0.684

of files belonging to tasks to decrease; this indicates that related files that do not include keywords cannot be found by either tasks or intertask relationships. In addition, we notice a trend of the F-measure decreasing when the $TransactionTime$ increases. This is because, as $TransactionTime$ increases, more related files are put into the same transaction, and the number of files in a set of transactions will be less than $MinSuppCnt$ because the number of accesses is only counted once during the $TransactionTime$-long interval. We omit the details of the experimental results for each dataset because of space limitations. Analysis of the results indicates that $TransactionTime = 3600$ accomplished the highest F-measure on datasets A, B and E, while $TransactionTime = 900$ showed the best performance on datasets C, D, F and G. Differences in work patterns can be inferred from these results. Based on the results, we used a different $TransactionTime$ in subsequent experiments.

Parameter Tuning for RMC Task Mining. $RMCTaskTime$ is the parameter used in extracting RMC tasks. To determine the value, in addition to FI tasks, we use the RMC tasks extracted in each case of $RMCTaskTime = \{60, 180, 300, 600, 1800\}$ [s] to perform file search.

As shown in Table 3, the average recall increased as $RMCTaskTime$ increased. However, the average precision decreased slightly, because a long $RMCTaskTime$ potentially brings unrelated files into the same RMC tasks. As the RMC operations were not handled very frequently, the impact was small and caused little change in the F-measure. In subsequent experiments, $RMCTaskTime$ was set to 60 s.

Table 4. Setups for Comparison of RMC Operations

Setup	Rename		Move		Copy	
	α_1	α_2	β_1	β_2	γ_1	γ_2
non-RMC	0	0	0	0	0	0
Rename	1	1	0	0	0	0
Move	0	0	1	1	0	0
Copy	0	0	0	0	1	1
RMC	1	1	1	1	1	1

4.4 Evaluation of Intertask Relationships

Relationships between tasks are derived from similarity links and RMC links. In this section, we conduct experiments to determine the appropriate values of parameters used in the formulas defined for calculating the weights of links and the degree of intertask relevance.

Experiments for RMC Links. RMC links, an aspect used for determining intertask relevance, are weighted using Equation (6), which considers the type and direction between files, by introducing rmc_f. rmc_f is defined by Equation (2) and the six parameters (α_1, α_2, β_1, β_2, γ_1, γ_2). To compare RMC operations, we set these parameters according to the configuration given in Table 4. As the results depicted in Fig. 5 show, using the rename and move operations had little effect on expanding the results of traditional file search. One reason is that renamed and moved files were ignored because files that do not exist in the filesystem were outside the evaluation targets. Another reason is that only 2% of log entries are created from rename and move operations. In contrast with rename and move, the copy operation enhanced the value of recall and the F-measure of full-text search results. Hence the result which utilized RMC performed the best F-measure, we set $(\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2) = (1.0, 1.0, 1.0, 1.0, 1.0, 1.0)$ in subsequent experiments.

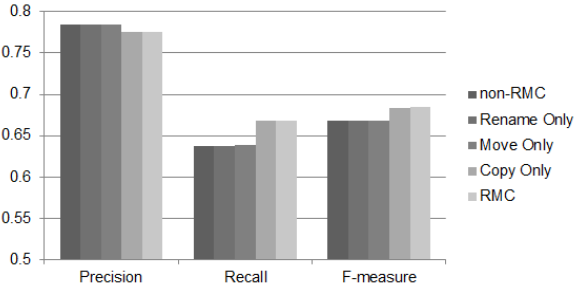


Fig. 5. Comparison of Experimental Results on RMC Links

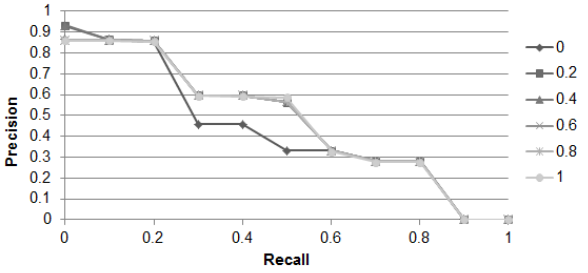


Fig. 6. Experimental Results for θ

Table 5. 11-point Average Precision

θ	0.0	0.2	0.4	0.6	0.8	1.0
Average of 11-point Average Precision	0.430	**0.482**	**0.482**	0.476	0.476	0.475

Experiment with θ. In Equation (7), we use θ to adjust the relative emphasis on similarity links and RMC links when calculating the degree of intertask relevance. To investigate the proper value of θ, we average the 11-point average precision with $\theta = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$.

As shown in Fig. 6, better-ranking results were obtained when $\theta > 0.0$ than when $\theta = 0.0$, although the precise value of θ was not so important. Because tasks that have more links to other tasks tend to obtain a higher score under the proposed method, even if the degree of intertask relevance changes, tasks of this nature would be prioritized. $\theta = 0.0$ led to a poor performance because using RMC links only was not sufficient to expand the full-text results to other tasks. We also note that $\theta = \{0.2, 0.4\}$ got the highest 11-point average precisions (Table 5).

4.5 Evaluation of SUGOI

To improve the results of an existing keyword-based search engine, we group files related to the same tasks and derive intertask relationships from access frequency and RMC operations on files. We propose methods for mining FI tasks and RMC tasks. To expand the full-text search results, we use similarity links and RMC links to determine the relevance between tasks. To investigate the effect of task mining and the association links, we use the setup shown in Table 6 to compare SUGOI with an existing full-text search engine [4].

The experimental results are given in Table 7 and Fig.7. All configurations of the proposed system SUGOI are better according to the F-measure than the existing full-text search engine. FI tasks and similarity links increased the average of recall from 0.273 to 0.5, while precision increased from 0.795 to 0.820 (SUGOI 1). In general, there is a trade-off between precision and recall. However,

Table 6. Setup of SUGOI

Setup	Task Type	Intertask Relational Links
SUGOI 1	FI Task Only	Similarity Links Only
SUGOI 2	FI Task Only	Similarity Links + RMC Links
SUGOI 3	FI Task + RMC Task	Similarity Links Only
SUGOI 4	FI Task + RMC Task	Similarity Links + RMC Links

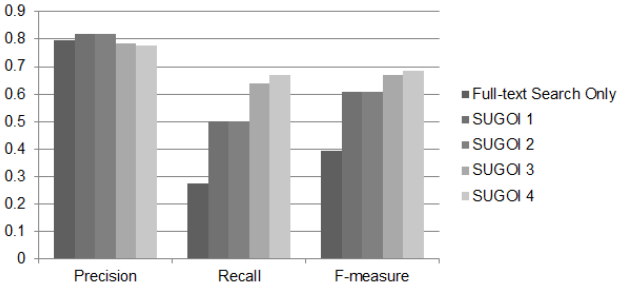


Fig. 7. Experimental Results of SUGOI

FI tasks consist of groups of files that are frequently accessed together and the average size of FI tasks is small, so few nonrelevant files are mingled in FI tasks.

In comparing SUGOI 1 with SUGOI 2 and SUGOI 3 with SUGOI 4, without RMC tasks, the effect of RMC links was small. An analysis of the mining results showed that only about 9% of log entries were RMC entries and most RMCed files were not contained in FI tasks because of low access frequency. Therefore, it was difficult to find new FI tasks using RMC links.

In addition to FI tasks and similarity, using RMC tasks and RMC links resulted in a certain improvement in recall and F-measure (SUGOI 3, 4). The reason is that files that do not include the keywords were found because they are related to tasks with files that do include the keywords. Thus, the precision average decreased slightly because RMC tasks are extracted from files RMCed together and, in contrast with FI tasks, extraneous files are easily mingled in RMC tasks.

From Table. 7, it is clear that SUGOI 4 achieved the highest recall and F-measure. The result confirmed that our proposed methods for task mining and deriving the intertask relationships are significantly effective for file search.

4.6 Summary of Experiments

In this section, we inspected the characteristics of the parameters and confirmed the validity of SUGOI by using actual file-access logs. The following conclusions were obtained.

Table 7. Experimental Results of SUGOI

Setup	Precision	Recall	F-measure
Traditional Full-text Search	0.795	0.273	0.392
SUGOI 1	0.820	0.500	0.606
SUGOI 2	0.820	0.500	0.606
SUGOI 3	0.784	0.638	0.668
SUGOI 4	0.776	0.669	0.684

1. A lower minimal support count ($MinSuppCnt = 2$) generated results with a higher F-measure. This means that if a combination of file accesses occurs more than twice in a set of transactions, these files are likely to have relevance to each other.
2. In the experiment on RMC task mining, we observed the trend that increasing $RMCTaskTime$ decreases the precision and increases the recall. However, the effect is small because users do not perform RMC operations frequently.
3. Copy operations, which can be done without making changes to the original file, were the most effective in RMC.
4. By investigating θ, a parameter used in Equation (7), we found that emphasizing the RMC links is effective, but the similarity links were also essential. SUGOI generated the best-ranking results when $\theta = \{0.2, 0.4\}$.
5. SUGOI conspicuously improves the average recall and F-measure over traditional full-text search results. The recall rise represented an increase of 0.396, while the increase in the F-measure was 0.292.

5 Conclusion and Future Work

As the volume of data stored in filesystems is increasing rapidly, and a large proportion of this is file-based unstructured data such as multimedia files, many files cannot be found using traditional full-text search engines. In this paper, we proposed a method for searching for such files by introducing the concept of tasks and intertask relationships derived from file-access logs. The main contributions of this paper are summarized as follows.

1. Task mining methods for two types of task: FI tasks, consisting of files frequently accessed together, and RMC tasks, consisting of files that were renamed/moved/copied (RMCed) simultaneously.
2. Methods for deriving association links from file-access logs by considering the similarity and RMC operations between tasks to generate a graph of intertask relationships.
3. A search method incorporating the task mining results into a full-text search to accomplish an accurate keyword-based file search.
4. Experimental results using actual file-access logs, which demonstrated that the proposed approach significantly improves search results.

As future work, we plan to conduct evaluation experiments using larger file-access logs. Contriving measures to choose the parameters automatically is also important for practical use. We also want to refine the proposed methods of task mining and the formulas for indicating intertask relevance.

Acknowledgments. This research was supported in part by a MEXT Grant-in-Aid for Scientific Research on Priority Areas (#201013017) and a JSPS Grant-in-Aid for Scientific Research (A) (#22240005).

References

1. Apple Inc.: Spotlight, http://www.apple.com/macosx/what-is-macosx/spotlight.html
2. Chen, J., Guo, H., Wu, W., Wang, W.: iMecho: an associative memory based desktop search system. In: CIKM 2009: Proceeding of the 18th ACM Conference on Information and Knowledge Management, pp. 731–740. ACM, New York (2009)
3. Google: Google Desktop, http://desktop.google.com
4. Hirabayashi, M.: Hyper Estraier, http://fallabs.com/hyperestraier/
5. Microsoft Corporation: Windows Search, http://www.microsoft.com/windows/products/winfamily/desktopsearch/
6. Daikoku Net: FAccLog, http://www2s.biglobe.ne.jp/~masa-nak/faldown.htm
7. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Tech. rep. Stanford University (1998)
8. Soules, C.A.N., Ganger, G.R.: Connections: using context to enhance file search. SIGOPS Oper. Syst. Rev. 39(5), 119–132 (2005)
9. Watanabe, T., Kobayashi, T., Yokota, H.: A Method for Searching Keyword-Lacking Files Based on Interfile Relationships. In: Chung, S., Herrero, P. (eds.) OTM-WS 2008. LNCS, vol. 5333, pp. 14–15. Springer, Heidelberg (2008)
10. Watanabe, T., Kobayashi, T., Yokota, H.: Searching Keyword-lacking Files Based on Latent Interfile relationship. In: Software and Data Technologies, pp. 236–244 (2010)
11. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New Algorithms for Fast Discovery of Association Rules. In: KDD-1997 Proceedings, pp. 283–286 (1997)

A General Top-k Algorithm for Web Data Sources

Mehdi Badr and Dan Vodislav

ETIS, CNRS, University of Cergy-Pontoise, France
firstname.lastname@u-cergy.fr

Abstract. Several algorithms for top-k query processing over web data sources have been proposed, where sources return relevance scores for some query predicate, aggregated through a composition function. They assume specific conditions for the type of source access (sorted and/or random) and for the access cost, and propose various heuristics for choosing the next source to probe, while generally trying to refine the score of the most promising candidate. We present *BreadthRefine* (BR), a generic top-k algorithm, working for any combination of source access types and any cost settings. It proposes a new heuristic strategy, based on refining all the current top-k candidates, not only the best one. We present a rich panel of experiments comparing BR with state-of-the art algorithms and show that BR adapts to the specific settings of these algorithms, with lower cost.

Keywords: top-k queries, web data, multi-criteria information retrieval, ranking.

1 Introduction

Data retrieval applications accorded an increasing importance these last years to ranked queries, compared to traditional boolean queries. This is, to a large extent, the consequence of the great development of web applications integrating huge volumes of heterogeneous and multimedia data: text, images, video, maps, etc. On one hand, this introduced the need for fuzzy, approximate answers, ranked by relevance, when querying such large and heterogeneous amount of data, on the other hand the new web data types came with intrinsic ranked predicates: text and image similarity, location proximity, user preferences, etc.

In this context, we address the problem of processing top-k queries over a set of web sources. A query is expressed through a set of ranked predicates over a common set of data objects. Each predicate is independently evaluated by some web source and returns a relevance score for any input object. A monotone aggregation function combines partial scores from each predicate into the final object score relative to the query. The query returns objects having the k best global scores.

Consider the example of a tourist in Paris, looking for buildings constructed around 1750, near to his current location and similar to the one he just photographed. Objects are here the buildings in Paris and the query consists of three

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 379–393, 2011.

```
select ∗ from Building b
order by proximity(b.year, 1750) +
         closeness(b.address, here()) +
         similarity(b.image, myImage)
limit 5 ;
```

S_1 (S-source)	S_2 (SR-source)	S_3 (R-source)
(o_2, 0.4)	(o_3, 0.9)	(o_1, 0.9)
(o_1, 0.3)	(o_1, 0.2)	(o_2, 0.7)
(o_4, 0.25)	(o_4, 0.15)	(o_3, 0.8)
(o_3, 0.2)	(o_2, 0.1)	(o_4, 0.6)

Fig. 1. Example query and sources

ranked predicates: p_1: proximity of the construction date, p_2: spatial proximity to the current location, and p_3: similarity with a given picture.

In SQL-like syntax, e.g. the one proposed by RankSQL [10], this query could be expressed such as in Figure 1, if we consider $k = 5$ and a simple aggregation function based on the sum of the individual scores.

The sources that evaluate the predicates could be, for instance:

- S_1: a database server storing historical information about buildings (for p_1);
- S_2: a geographic/map service (for p_2);
- S_3: an image database indexing fronts of buildings in Paris (for p_3).

Note that the ranking predicates are *dependent on the query*, for instance, even if S_2 always evaluates spatial proximity between an object and a reference point, this point depends on the query.

To execute such a query, one must access the web sources to get partial scores for objects. The access to the web sources during query processing has the following main properties:

- Access is limited to operations allowed by the web source interface and there is no control on the inside mechanisms. Generally, a web source may allow two kinds of access: *sorted*, where each access returns the next score/object in decreasing order, and/or *random*, where each access returns the score of a given object. We call *S-sources* sources with sorted access only, *R-sources* those with random access only, and *SR-sources* those with both accesses.
- Accessing web sources is expensive, the cost of accessing sources dominates the cost of the other algorithm operations.

The naive approach would be to get all the partial scores for all the objects, then to compute their global scores, to order them by descending score and get the first k results. In practice, this method is very expensive and many efficient algorithms have been proposed for various cases of access types and cost settings.

Some algorithms, such as NRA [4] and StreamCombine [6] consider only S-sources, while algorithms such as TA [4], CA [4] and QuickCombine [5] consider only SR-sources. A third category considers specific heterogeneous access configurations, e.g. one S-source and several R-sources for Upper [2] and MPro [3], or several SR-sources and several R-sources for some extensions of TA and Upper. The NC framework [14] is the only approach that addresses any combination of access types. All these algorithms are further detailed in the related work section.

Cost settings considered by these algorithms fall generally into two categories: (i) no difference between the cost of sorted and random accesses (NRA,

StreamCombine, TA, QuickCombine), and (ii) random more expensive than sorted access (CA, Upper, MPro). However, other cost settings are possible, e.g. sorted more expensive than random access. Consider for instance the case of a source evaluating image similarity by using an index on disk for sorted access, but keeps image descriptors in memory; random access requires here no disk access and is faster than sorted access. Again, NC is the only attempt to adapt to any cost settings.

The general idea behind all these top-k algorithms is to maintain a list of candidate objects and the interval $[L, U]$ of possible global scores for each of them. At the beginning, the interval is obtained by aggregating the minimum / maximum source scores. Monotonicity of the aggregation function ensures that further source accesses will refine these intervals, by decreasing the upper bound U and increasing the lower bound L. The algorithm stops when the score of the best k candidates cannot be exceeded by the other objects.

We illustrate the behavior of such an algorithm through the example in Figure 1. Suppose S_1 is a S-source, S_2 a SR-source, S_3 a R-source; scores are presented in descending order for S/SR sources and by object id for R-sources. Individual predicate scores belong to the $[0, 1]$ interval, so the initial global score interval is $[0, 3]$ for all objects. Note that we consider *algorithms without wild guesses*, i.e. objects are not known in advance, they are 'discovered' by sorted accesses. We note *candidates* the set of candidates and U_{unseen} the maximum score of objects not yet discovered. Initially, $candidates = \emptyset$ and $U_{unseen} = 3$.

- A sorted access to S_1 retrieves $(o_2, 0.4)$, so with one partial score known for o_2 its global score interval becomes $[0.4, 2.4]$, i.e. $candidates$=$\{(o_2, [0.4, 2.4])\}$. Also U_{unseen} becomes 2.4 because further scores in S_1 cannot exceed 0.4.
- A sorted access to S_2 retrieves $(o_3, 0.9)$. This adds a new candidate (o_3), lowers U_{unseen} to 2.3 (because further S_2 scores cannot exceed 0.9), but also lowers the maximum global score of o_2 to 2.3, because the maximum score of S_2 is now 0.9. $candidates$=$\{(o_2, [0.4, 2.3]), (o_3, [0.9, 2.3])\}$.
- A random access to S_2 for o_2 retrieves $(o_2, 0.1)$. This changes only the global score interval of o_2. $candidates$=$\{(o_2, [0.5, 1.5]), (o_3, [0.9, 2.3])\}$.
- A random access to S_3 for o_3 retrieves $(o_3, 0.8)$ and changes the global score interval of o_3. $candidates$=$\{(o_2, [0.5, 1.5]), (o_3, [1.7, 2.1])\}$.
- A sorted access to S_2 retrieves $(o_1, 0.2)$. This adds a new candidate (o_1), lowers U_{unseen} to 1.6, but does not lower the maximal global score of the other candidates because o_2 and o_3 already know their S_2 scores. $candidates$=$\{(o_2, [0.5, 1.5]), (o_3, [1.7, 2.1]), (o_1, [0.2, 1.6])\}$.
- The minimum global score of o_3 exceeds both U_{unseen} (maximum global score of unseen objects) and the maximum global score of all the other candidates, we can conclude that o_3 is the best (top-1) object.

Execution may continue depending on the value of k, but notice that *top-k objects could be returned without knowing their exact scores and order*. Most of the existing algorithms return exact scores for top-k and are iterative, i.e. return first top-1, then top-2, etc. However, in many applications the exact global score is not needed (e.g. image retrieval), but only the top-k objects with approximate ordering.

The difference between the top-k algorithms mentioned above consists in the heuristics proposed for the choice of the next source to probe and of the candidate to refine through random accesses. Note that all the algorithms that must select a candidate to further refine focus on the current "best" candidate.

Our goal is to propose a new generic top-k algorithm that works for any combination of source access types and cost settings, and returns the set of top-k objects, possibly without complete scoring information. This algorithm uses a new heuristic approach, that refines on all the top-k candidates, instead of favoring only the best current candidate.

The contributions of this paper may be resumed as follows:

- We present *BreadthRefine* (BR), a new, generic top-k algorithm, able to adapt to any combination of source access types and to any cost settings. To the best of our knowledge, excepting NC [14], this is the first generic top-k algorithm for web sources. Unlike NC, that mixes heuristics and sampling optimization, our algorithm is only based on a simple heuristics.
- We propose a new heuristic approach, that do not focus only on the best current candidate, but considers all the top-k candidates. Our experiments show that this heuristics produces better results than the usual approach.
- We report a rich experimental evaluation comparing BR to existing algorithms for various source types and cost settings, and show that BR is less expensive.

The rest of the paper is organized as follows: the next section presents related work, then Section 3 describes the BR algorithm, Section 4 reports the experiments comparing BR to existing algorithms, then we end with conclusions and future work.

2 Related Work

Top-k query processing techniques have been largely studied in the last decade at different levels: query model, access types, implementation structures, integration in database engines [10] [7] [9], etc. The survey [8] presents a rich overview of these various approaches. In this context, we address the problem of selection queries (no joins), for web sources with any configuration of access types and any cost settings. We do not compare with algorithms having additional information about objects in sources, e.g. the rank in BPA [1]. Join queries are addressed e.g. by the J* [13] or the Rank-Join [7] algorithms.

Algorithms for top-k selection queries proposed so far focus on specific access types and cost settings for the sources. They fit with the general method illustrated in the example of Section 1, i.e. maintaining a list of candidates with global score interval and accessing sources following their own heuristics.

Among the algorithms that consider only S-sources, the best known is *NRA* (No Random Access) [4]. NRA consults sources following a simple round-robin strategy, with no specific order. Such as for BR, final results may have incomplete scoring information. Efficient NRA implementations such as LARA [11] that reduce the overhead of candidate updates are not relevant in our context, since

we ignore this overhead compared to the access time to web sources. *StreamCombine* [6] is a variant of NRA that selects at each step the next source to probe following the benefit that the source may provide. Benefit considers three factors: (i) the importance of the source in the global score (e.g. the coefficient in the aggregation function, for a weighted sum), (ii) the decrease of the source score (bigger decreases favor faster algorithm termination), and (iii) the number of candidates in the current top-k not yet seen in the source and that will consequently lower their upper bound. We adopt in BR a similar notion of benefit for sorted sources.

Algorithms that consider only SR-sources adopt a different approach: the global score of a candidate discovered through a sorted access is completely evaluated through random probes of the other sources. The consequence is that candidates have exact scores, not intervals, so there is no need to maintain a list of candidates. The termination test simply compares the score of the k-th candidate with the threshold U_{unseen}.

TA (Threshold Algorithm) [4], the best known SR-source algorithm, consults sources in a way similar to NRA, following a round-robin strategy. *QuickCombine* [5] is a variant of TA that uses the same idea as StreamCombine to select the next sorted source to probe. The benefit considers only the two first factors presented above for StreamCombine, the last one being not relevant. *CA* (Combined Algorithm) [4] is a variant of TA that considers random accesses being h times more expensive than sorted ones. It combines NRA with TA to reduce the number of random accesses by performing h sorted accesses in each source before a complete evaluation of the best candidate by random probes. We adopt in BR a similar idea for taking into account the possible difference between random and sorted costs. Many other extensions of TA have been proposed, such as *TAz* [2], which considers an additional set of R-sources. TAz acts as TA for the sorted accesses, but includes the R-sources in the random probe phase.

Algorithms with sorted access and controlled random probes, such as *Upper* [2] and *MPro* [3], typically consider one S-source and several R-sources. The random cost exceeds the sorted cost, but is different from one R-source to another. Both Upper and MPro consider complete scoring of the final top-k result.

Upper maintains candidates following an *estimated* score. At each step, it considers candidate o with the highest upper score U: if $U < U_{unseen}$ a sorted access is performed in order to reduce U_{unseen}, otherwise a random probe for o is done. A benefit is computed for each R-source and the best source is selected for the random probe. Two cases are considered. If o belongs to the current top-k, the benefit is the ratio between the expected decrease δ of U and the access cost. Otherwise, o has more chances to be out of the top-k and one computes the decrease Δ of U necessary to prove that o is not a top-k object. The source benefit in this case is the ratio between $min(\delta, \Delta)$ and the access cost. An extension of Upper for several SR-sources and several R-sources is presented in [12] - in this case sorted access is performed in NRA manner over the SR-sources, while random access considers all the SR/R-sources not yet probed.

Like Upper, *MPro* considers at each step the candidate with the highest upper score and performs a random probe for it or a sorted access if U_{unseen} is higher. But unlike Upper, MPro fixes for all the candidates *the same order* (schedule) for probing the R-sources. The best schedule may be determined by various methods, such as sampling optimization, proposed by the authors.

The only algorithm that aims at genericity is *NC* (Necessary Choices) [14], an extension of MPro. Like BR, NC is generic and adapts to any source access type and cost settings. NC identifies *necessary choices* (i.e. accesses that are necessary at a given execution state to obtain the final result) as belonging to accesses for the current top-k upper bound candidates. NC proposes an algorithm framework that performs only necessary accesses and defines an algorithm called SR/G in this framework that computes for each S/SR-source a *limit score*. SR/G gives priority to sorted accesses in sources that did not reach the limit score. More precisely, the algorithm considers at each step the candidate with the highest upper score and chooses a sorted access for it, if possible (i.e. in a sorted source that did not return the object, nor reach its limit). If not possible, a random probe is selected following a fixed schedule, like for MPro. Limit scores are determined by sampling optimization and simulation.

Compared to NC, BR addresses a slightly different problem, by computing top-k objects possibly without complete scoring. Therefore necessary choices in the NC context are not relevant for BR. We do not compare in this paper BR with NC because, source sampling being not always possible, we only focus on fully heuristic-based top-k algorithms. Even if NC authors claim that sampling may replaced by estimation of the source limits, our tests indicate that NC is very sensible to the quality of this estimation. One should first define good limit estimation heuristics for NC, which is out of the scope of this paper.

3 The *BreadthRefine* Algorithm

Unlike top-k algorithms presented in Section 2, the *BreadthRefine* (BR) algorithm covers any configuration of source access types and cost settings and proposes a new heuristic approach: refining the scores of all the top-k candidates, instead of focusing on the best one. We first present the BR data and query model, then the general BR algorithmic framework and several algorithm variants in this framework.

Data and query model. We consider a set of data objects $\mathcal{O} = \{o_1, ..., o_n\}$, a top-k multi-criteria query q over these objects, and a set of web sources $\mathcal{S} = \{S_1, ..., S_m\}$ able to evaluate the query criteria.

Definition 1. ***Query:*** A top-k multi-criteria query q *is defined by (i) a number of objects* k *to return, (ii) a set of ranked predicates (criteria)* $P_q = \{p_1, ..., p_m\}$*, depending on the query, and (iii) a monotone score aggregation function* $\mathcal{F}$*.*

Predicates $p_j : \mathcal{O} \rightarrow [min_j, max_j]$ *return for any object a score in a given interval, while function* $\mathcal{F} : \mathbb{R}^m \rightarrow \mathbb{R}$ *aggregates predicate scores into a global object score. Each predicate* p_j *is independently evaluated by source* S_j*.*

Definition 2. ***Source:*** A source S_j *is characterized by (i) its access type (S, R or SR), and (ii) a cost per access, noted* $C_s(S_j)$ *for sorted and* $C_r(S_j)$ *for random access. The minimum and maximum scores in* S_j *are noted* $\min_j$ *and* $\max_j$*, as mentioned in Definition 1.*

The set of sources $\mathcal{S}$ *can be partitioned following the access type in three disjoint subsets, possibly empty:* $\mathcal{S} = \mathcal{S}_S \cup \mathcal{S}_R \cup \mathcal{S}_{SR}$.

A sorted source (of type S or SR) provides an access function

$$\text{getNext} : \mathcal{S}_S \cup \mathcal{S}_{SR} \rightarrow \mathcal{O} \times \mathbb{R} \cup \{\text{nil}\}$$

returning the next couple (o, s), where o is the object with the next score in decreasing order and s its score (if exists), or the special value nil *otherwise.*

A random source (of type R or SR) provides an access function

$$\text{getScore}:(\mathcal{S}_R \cup \mathcal{S}_{SR}) \times \mathcal{O} \rightarrow \mathbb{R}$$

returning the score of the given object in the source.

We note score(o, S_j) *the score retrieved for object o in source* S_j.

We note crtmax_j *the largest score that source* S_j *could further return. For pure random sources* $S_j \in \mathcal{S}_R$, $crtmax_j = max_j$ *(constant). For sorted sources* $S_j \in \mathcal{S}_S \cup \mathcal{S}_{SR}$, crtmax_j *is the score returned by the last sorted access to* S_j[1] *(initially* $crtmax_j = max_j$*).*

Definition 3. ***Candidates:*** A candidate *in the algorithm is an object that has been already returned by a sorted access. For each candidate c, the algorithm maintains* $L(c)$ *(*$U(c)$*), i.e. the lower (upper) bound of the global score for c.* $L(c)$ *(*$U(c)$*) is computed by aggregating the scores of c in the sources where it has been already consulted, and the minimum (maximum) score in the other sources.*

$L(c) = \mathcal{F}(l_1, ..., l_m)$, l_j =score(c, S_j) *if c consulted in* S_j, *else* $l_j = min_j$

$U(c) = \mathcal{F}(u_1, ..., u_m)$, u_j =score(c, S_j) *if c consulted in* S_j, *else* $u_j = crtmax_j$.

We note L_k *(*U_k*) the k-th value in decreasing order for* $L(c)$ *(*$U(c)$*) among the candidates - for less than k candidates, the value is* nil.

A candidate is called viable *if it still has chances to belong to the final top-k result. The viability condition for c is* $U(c) \geq L_k$. *It is simple to prove using monotonicity that once a candidate becomes non-viable, it will remain non-viable and could be removed.*

We note U_{unseen} *the upper bound of the global score for any object that is not yet a candidate. Initially,* $U_{unseen} = \mathcal{F}(max_1, ..., max_m)$

The BR algorithm framework. The basic idea of the BR algorithm is to maintain the top-k candidates as a whole instead of looking only at the best candidate. Also, BR is able to handle any kind of source access; the choice of the access type is the central issue at each step.

Figure 2 presents the general BR framework, from which various algorithm variants may be instantiated. BR maintains a set of candidates, initially empty, and the maximum score of unseen objects, U_{unseen}.

[1] We consider that an object retrieved through a sorted access in a SR-source is not further accessed by a random access in the same source.

Framework $BR(q, \mathcal{S})$
  $candidates \leftarrow \emptyset$
  $U_{unseen} \leftarrow \mathcal{F}(max_1, ..., max_m)$
  repeat
    //choice between sorted or random access
    if $|candidates| < k$ **or** $U_k < U_{unseen}$ **or** ***CostCondition()*** **then**
      $S_j \leftarrow$ ***BestSortedSource(*** $\mathcal{S}$ ***)*** *//choice of a sorted source*
      $(o, s) \leftarrow getNext(S_j)$ *//sorted access to the selected source*
      Update *candidates* and U_{unseen}
    else *//random access*
      $c \leftarrow$ ***ChooseCandidate(*** **candidates,** k ***)*** *//choice of a top-k candidate*
      $S_j \leftarrow$ ***BestRandomSource(*** $\mathcal{S}$**,** c ***)*** *//choice of a random source*
      $s \leftarrow getScore(S_j, c)$ *//random access to the selected source*
      Update *candidates*
    endif
  until $|candidates| = k$ and $L_k \geq U_{unseen}$
  return *candidates*

Fig. 2. The BR algorithm framework

At each step, BR first chooses the type of access to be performed. Sorted access is preferred in the following cases:

- A group of top-k candidates does not exist yet, i.e. if $|candidates| < k$.
- An unseen object could belong to the current upper bound top-k group, i.e. $U_k < U_{unseen}$. A sorted access will decrease U_{unseen} and eliminate unseen objects from the top-k group.
- If the cost-related condition *CostCondition* is true. This condition enables BR to adapt to various cost settings, e.g. to force sorted accesses when random probes are expensive.

If a sorted access is decided, the *BestSortedSource* function chooses the best sorted source, then performs the sorted access[2]. Consequently, the candidate set and U_{unseen} are updated; the update of the candidate set consists in several actions:

- Add the object to the candidate set, if it is seen for the first time.
- Update the upper and lower bounds for all the candidates. In fact, besides the retrieved object, the update only affects the upper bound of candidates that have not been retrieved yet in the source.
- Remove non-viable candidates.

In the case of a random access, the *ChooseCandidate* function chooses a candidate in the current top-k group. Then a random source is selected with *BestRandomSource*, among those not yet probed for the candidate. The source is probed and the candidate set is updated.

The algorithm ends when the candidate set is reduced to k objects (after removing non-viable candidates) and when the unseen objects cannot change

[2] We consider that *BestSortedSource* does not return a source with no more objects.

anymore the final result ($L_k \geq U_{unseen}$). The result is the set of k candidates, with possibly incomplete scores.

BR algorithm variants. By instantiating *CostCondition*, *BestSortedSource*, *ChooseCandidate* and *BestRandomSource* functions, one may obtain various BR algorithms. We present three variants: *BR-Cost*, *BR-Basic* and *BR-First*.

BR-Cost is the reference BR algorithm that we propose. It refines the set of top-k candidates in a breadth-first manner and adapts to various cost settings. *BR-Cost* uses weighted sum aggregation function and employs source selection methods similar to existing algorithms.

- *ChooseCandidate* selects the least refined candidate (with the least random probes) from the current upper bound top-k group.
- *CostCondition* aims at reducing the number of random accesses (if more expensive than sorted ones), in a way similar to the CA algorithm. More precisely, if r is the average ratio between the cost of random and sorted accesses, then once a random probe is processed, the next one is possible only after at least r sorted accesses.
- *BestSortedSource* selects the sorted source S_j with respect to a benefit similar to the one proposed by StreamCombine [6], i.e. $B_j = coef_j N_j \delta_j / C_s(S_j)$, where $coef_j$ is the weight of S_j in the aggregation function, N_j the number of top-k candidates not yet seen in S_j, δ_j the expected decrease of the score in S_j and $C_s(S_j)$ the access cost. Here N_j measures the number of top-k objects that will be concerned by the decrease of the upper bound, and $coef_j$ and δ_j the amount of this decrease.
- *BestRandomSource* selects the random source with respect to a benefit $B_j = coef_j \times (crtmax_j - min_j)/C_r(S_j)$. Here $coef_j$ and $crtmax_j - min_j$ measure the variation of the candidate's upper/lower bound when the real score will replace the upper/lower source score.

BR-Basic is a variant of *BR-Cost* which does adapt to various cost settings, i.e. *CostCondition* systematically returns *false*. Comparing BR-Basic with BR-Cost will reveal the importance of cost adaptation.

BR-First is a variant of *BR-Basic* that uses the classical approach for choosing a candidate to refine, i.e. always select the best one - the highest upper bound in this case. Note that this is still a particular case of the general BR heuristics, the best candidate belonging to the top-k. Comparing *BR-First* with *BR-Basic* will measure the benefit of the new heuristics.

4 Experiments

In this section we report the experimental evaluation of the BR algorithms over synthetic data. We evaluate genericity and adaptivity by comparing BR with state of the art algorithms in their specific access type and cost configurations. We also measure the importance of the BR heuristics and of cost adaptivity by comparing the BR algorithms variants.

Data sets. We generate synthetic score lists for each source, each list representing the predicate scores for a given query, as values in the [0,1] interval. Sources are independent and have similar data distribution. The sorted access cost C_s is the same for any source, idem for the random cost C_r. We consider three variants of data distribution:

- *Uniform*: values are uniformly distributed.
- *Gaussian*: values are generated from three overlapping Gaussian bells.
- *Zipfian*: values are generated from a Zipf function with 1000 distinct values and Zipfian parameter $z = 1$.

Parameters and default settings. All the experiments measure *the execution cost*, which is the cost of all the source accesses performed during execution. More precisely, if Ns_j (Nr_j) is the number of sorted (random) accesses to source S_j, then the cost of the algorithm is:

$$cost = \sum_{S_j \in \mathcal{S}_S \cup \mathcal{S}_{SR}} Ns_j C_s(S_j) + \sum_{S_j \in \mathcal{S}_R \cup \mathcal{S}_{SR}} Nr_j C_r(S_j) \quad (1)$$

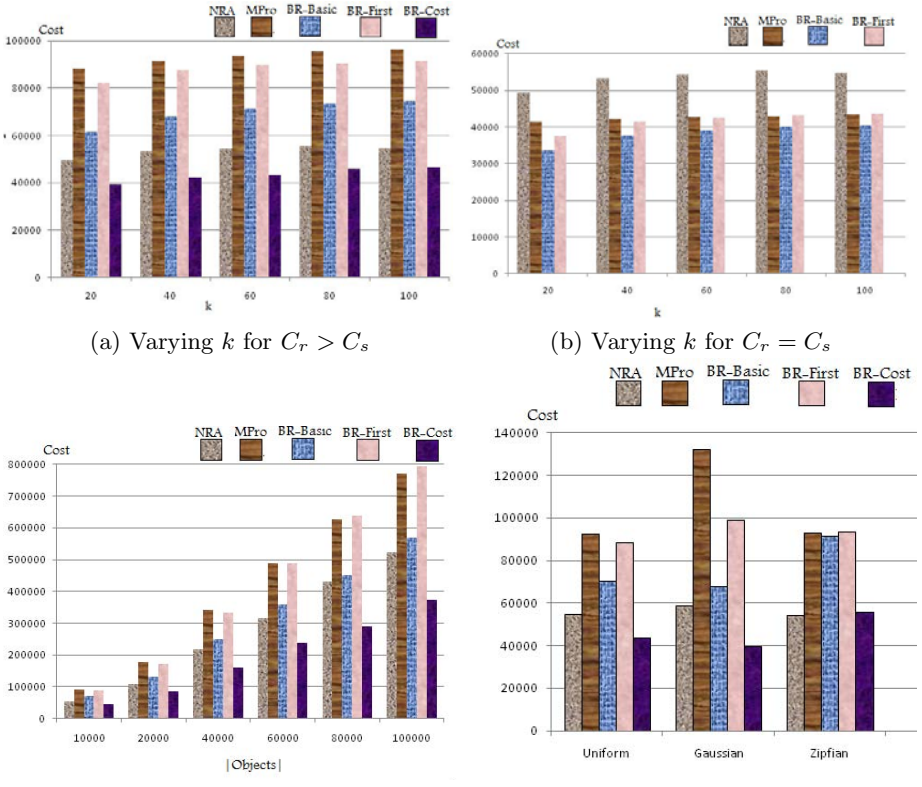
Each result is the average of 8 measures over different randomly generated data sources. We consider various parameters in the experiments:

- The number of objects: 10000 (default), 20000, 40000, 60000, 80000, 100000.
- The number k of returned objects: 20, 40, 50 (default), 60, 80, 100.
- The number of S-sources N_s, of R-sources N_r and of SR-sources N_{sr} (default value for each one: 3).
- The cost setting: $C_r(=5) > C_s(=1)$ (default), $C_r = C_s(=1)$, $C_r(=1) < C_s(=5)$
- Data distribution for all the sources: uniform (default), gaussian, zipfian.

Algorithms tested. Experiments are grouped by source access type in three categories, in each case the set of algorithms adapted to this setting is considered:

- No R-sources: NRA and MPro.
- No S-sources: Upper, TAz and MPro.
- All the source types exist: MPro.

We adapted the MPro algorithm to cover all the cases as follows: sorted sources are combined with NRA to behave as a single S-source. SR-sources may be considered by MPro either as sorted or random, depending on the context: sorted when there are no S-sources, random when there are no R-sources. In the case of all the source types, we considered both settings (SR as sorted and SR as random) and reported the average cost. Also for random probes we considered a fully heuristic variant for MPro, in which scheduling uses the same order based on benefit as BR.



(a) Varying k for $C_r > C_s$

(b) Varying k for $C_r = C_s$

(c) Varying the number of objects

(d) Varying the distribution

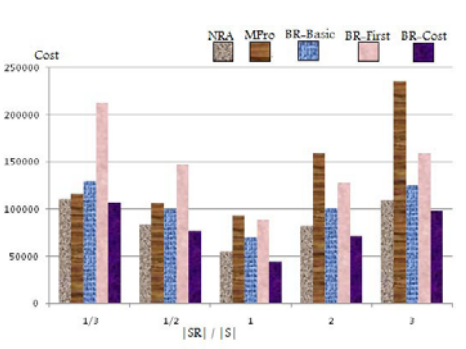
Fig. 3. SR + S-sources: varying k, the number of objects and the distribution

SR-sources + S-sources. We compare BR-Cost, BR-Basic and BR-First with NRA and MPro. We first vary k in two cases: $C_r > C_s$ (default setting) and $C_r = C_s$. The case $C_r < C_s$ is not favorable to NRA and is considered later for MPro.

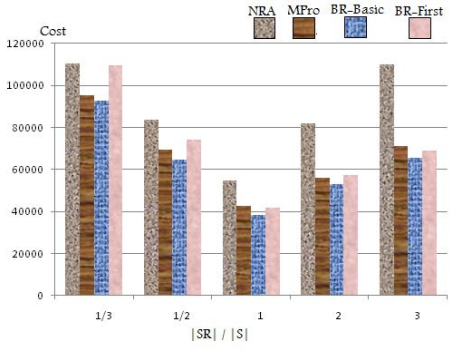
Figure 3(a) presents results for $C_r > C_s$. Cost increases with k for all the algorithms, but BR-Cost is significantly less expensive than MPro and even than NRA, which does no random access. The importance of considering cost settings in BR is obvious when comparing BR-Cost with BR-Basic. Also the benefit of the BR heuristic is confirmed by BR-First which behaves worse than BR-Basic. Figure 3(b) considers the case $C_r = C_s$, where BR-Cost is the same as BR-Basic. BR-Basic/Cost is still the best, but the difference with BR-First and MPro decreases.

Next, we vary the number of objects for the default cost settings. Figure 3(c) shows that the cost increases almost proportionally for all the algorithms, so the above conclusions are unchanged.

Figure 3(d) illustrates the variation of data distribution. For uniform and gaussian distributions, BR-Cost is the best algorithm and BR-Basic outperforms

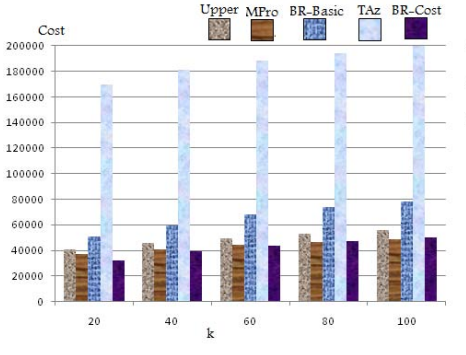


(a) $C_r > C_s$

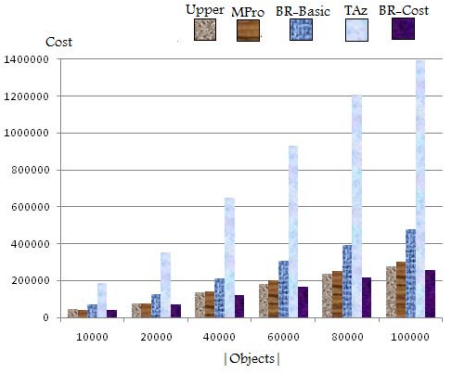


(b) $C_r = C_s$

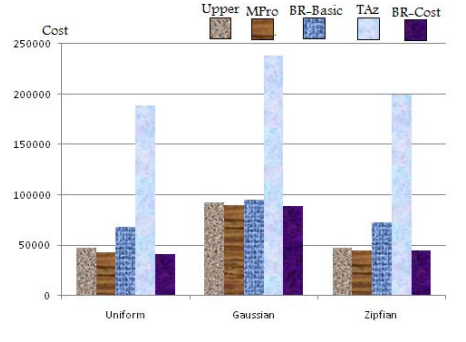
Fig. 4. SR + S-sources: varying |SR|/|S|



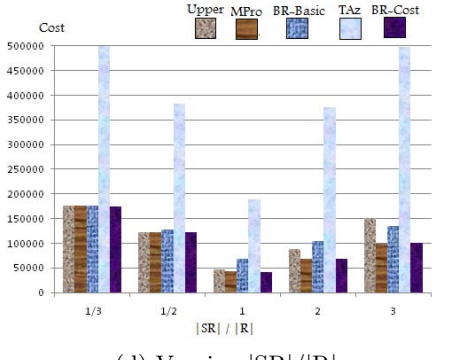
(a) Varying k



(b) Varying the number of objects



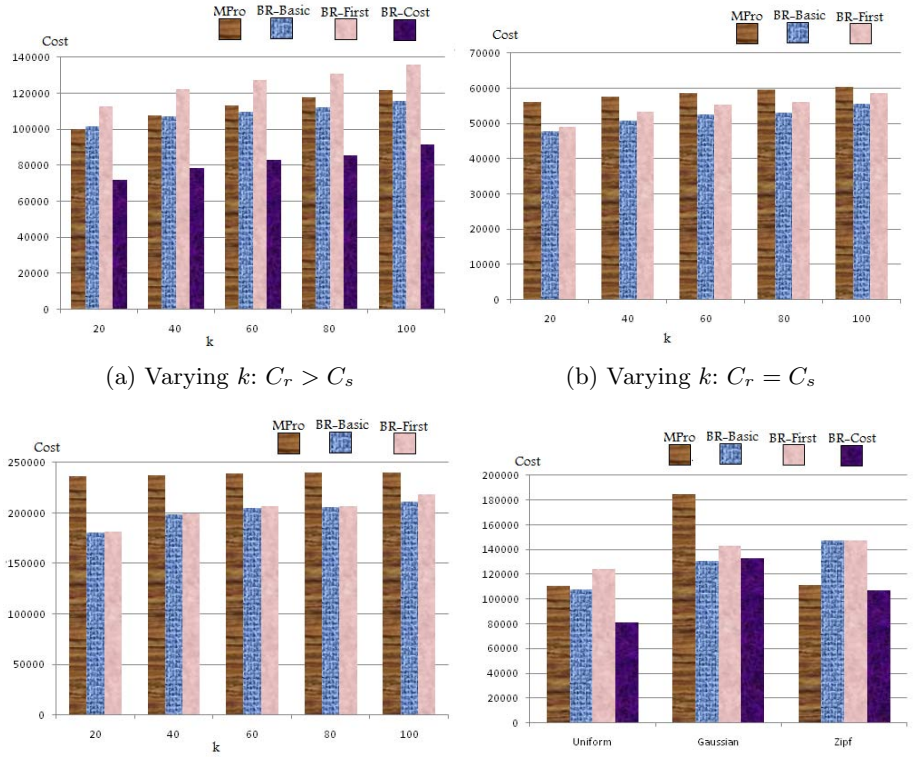
(c) Varying distribution



(d) Varying |SR|/|R|

Fig. 5. SR + R-sources

BR-First. Zipfian distribution, with many identical values, is a special case, but BR-Cost still have good results. BR-Cost and NRA have almost the same cost, while differences between MPro and BR variants become insignificant.



(a) Varying k: $C_r > C_s$

(b) Varying k: $C_r = C_s$

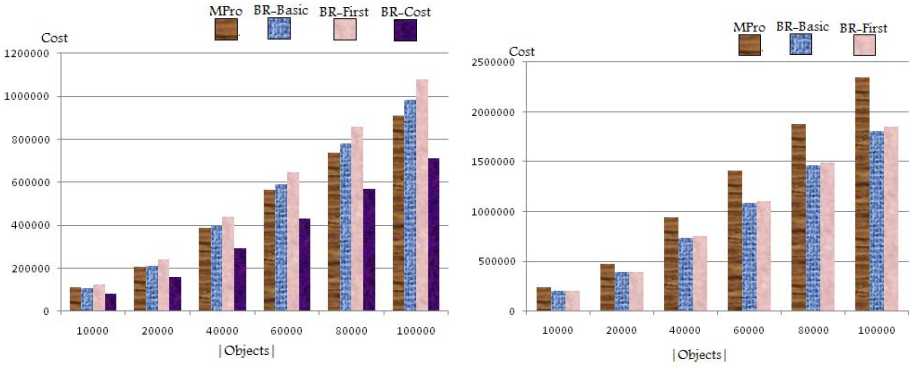
(c) Varying k: $C_r < C_s$

(d) Varying distribution

Fig. 6. SR + S + R-sources: varying k and the distribution

Finally, we study the impact of varying the ratio between the number of SR-sources and of S-sources. Figure 4 reports the results for the default setting (a) and for $C_r = C_s$ (b). Interestingly, in both cases algorithms have best results when |SR| = |S|. In the default setting, in all cases, BR-Cost has better cost, but it substantially outperforms the other algorithms when |SR| > |S|. Also BR-Basic is always significantly better than BR-First, the difference increases when |SR| < |S|. When $C_r = C_s$, BR-Basic/Cost, BR-First and MPro keep the same relative performances, while NRA behave worse when |SR| > |S|.

SR-Sources + R-Sources. We compare BR with Upper, TAz and MPro. For space reasons, we only illustrate BR-Cost and BR-Basic, knowing that measures indicate that BR-First remains worse than BR-Basic. Results are presented in Figure 5. TAz is systematically outperformed by all the other algorithms and BR-Basic by BR-First. BR-Cost has globally the best results, even if Upper and MPro are close. Upper is slightly worse than MPro in the default setting, but scales better than MPro when the number of objects grows. Also Upper degrades when |SR| > |R|.



(a) $C_r > C_s$ (b) $C_r < C_s$

Fig. 7. SR + S + R-sources: varying the number of objects

SR-Sources + S-Sources + R-Sources. We compare BR-Cost, BR-Basic and BR-First with MPro, by first varying k for all the cost settings. Figure 6(a) reports the default case $C_r > C_s$. BR-Cost outperforms MPro, while BR-Basic remains better than BR-First and similar to MPro. When $C_r = C_s$ (subfigure b), BR-Basic/Cost is better than both MPro and BR-First and the difference augments when $C_r < C_s$ (subfigure c). This seems to indicate that MPro makes more sorted accesses than BR-Basic (the difference increases when C_s grows) and probably less random accesses.

Figure 6 (d) illustrates the impact of data distribution: BR-Cost remains globally the best, while BR-First is the worse BR variant. There is little difference between BR variants for gaussian distribution, while zipfian favors MPro compared to BR-Basic and BR-First.

Figure 7 reports the impact of the number of objects for $C_r > C_s$ (a) and $C_r < C_s$ (b). In both cases, the cost augments almost proportionally for the BR variants and BR-Cost is always better than the other algorithms. We remark that MPro scales better than BR-Basic and BR-First when $C_r > C_s$ and significantly worse when $C_r < C_s$, indicating that MPro makes slightly less random accesses, but much more sorted accesses than BR-Basic.

5 Conclusion

In this paper we proposed *BreadthRefine* (BR), a generic top-k algorithm, able to adapt to any combination of source access types and to any cost settings. It adopts a new heuristic approach, by refining the scores of all the top-k candidates instead of focusing on the best one. Experiments on synthetic data clearly indicate that BR successfully adapts to various settings, with globally better execution cost than algorithms designed for that specific case. Comparison with the classical approach of favoring the best candidate shows that BR's breadth-first heuristics produces better results.

Future work will focus on the advantages of breadth-first heuristics in approximating the top-k final results.

References

1. Akbarinia, R., Pacitti, E., Valduriez, P.: Best position algorithms for top-k queries. In: VLDB, pp. 495–506 (2007)
2. Bruno, N., Gravano, L., Marian, A.: Evaluating top-k queries over web-accessible databases. In: ICDE (2002)
3. Chang, K.C.-C., won Hwang, S.: Minimal probing: supporting expensive predicates for top-k queries. In: SIGMOD Conference, pp. 346–357 (2002)
4. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci. 66(4), 614–656 (2003)
5. Güntzer, U., Balke, W.-T., Kießling, W.: Optimizing multi-feature queries for image databases. In: VLDB, pp. 419–428 (2000)
6. Güntzer, U., Balke, W.-T., Kießling, W.: Towards efficient multi-feature queries in heterogeneous environments. In: ITCC, pp. 622–628 (2001)
7. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K.: Supporting top-k join queries in relational databases. VLDB J. 13(3), 207–221 (2004)
8. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. 40(4) (2008)
9. Li, C., Chang, K.C.-C., Ilyas, I.F.: Supporting ad-hoc ranking aggregates. In: SIGMOD Conference, pp. 61–72 (2006)
10. Li, C., Chang, K.C.-C., Ilyas, I.F., Song, S.: Ranksql: Query algebra and optimization for relational top-k queries. In: SIGMOD Conference, pp. 131–142 (2005)
11. Mamoulis, N., Cheng, K.H., Yiu, M.L., Cheung, D.W.: Efficient aggregation of ranked inputs. In: ICDE, p. 72 (2006)
12. Marian, A., Bruno, N., Gravano, L.: Evaluating top-k queries over web-accessible databases. ACM Trans. Database Syst. 29(2), 319–362 (2004)
13. Natsev, A., Chang, Y.-C., Smith, J.R., Li, C.-S., Vitter, J.S.: Supporting incremental join queries on ranked inputs. In: VLDB, pp. 281–290 (2001)
14. won Hwang, S., Chang, K.C.-C.: Optimizing top-k queries for middleware access: A unified cost-based approach. ACM Trans. Database Syst. 32(1), 5 (2007)

Improving the Quality of Web Archives through the Importance of Changes*

Myriam Ben Saad and Stéphane Gançarski

LIP6, University P. and M. Curie,
4 place Jussieu 75005, Paris, France
{myriam.ben-saad,stephane.gancarski}@lip6.fr

Abstract. Due to the growing importance of the Web, several archiving institutes (national libraries, Internet Archive, etc.) are harvesting sites to preserve (a part of) the Web for future generations. A major issue encountered by archivists is to preserve the quality of web archives. One way of assessing the quality of an archive is to quantify its completeness and the coherence of its page versions. Due to the large number of pages to be captured and the limitations of resources (storage space, bandwidth, etc.), it is impossible to have a complete archive (containing all the versions of all the pages). Also it is impossible to assure the coherence of all captured versions because pages are changing very frequently during the crawl of a site. Nonetheless, it is possible to maximize the quality of archives by adjusting web crawlers strategy. Our idea for that is (i) to improve the completeness of the archive by downloading the most *important* versions and (ii) to keep the most important versions as coherent as possible. Moreover, we introduce a pattern model which describes the behavior of the importance of pages changes over time. Based on patterns, we propose a crawl strategy to improve both the completeness and the coherence of web archives. Experiments based on real patterns show the usefulness and the effectiveness of our approach.

Keywords: Web Archiving, Data Quality, Change Importance, Pattern.

1 Motivation

The main goal of web archiving institutes (national libraries, Internet Archive, etc.) is to preserve the history of web sites for future generations. Most often, web archiving is automatically performed using web crawlers. Web crawlers visit web pages to be archived and build a snapshot and/or index of web pages. In order to maintain the archive up-to-date and to preserve its quality, crawlers must revisit periodically the pages and update the archive with fresh images. We define the quality of an archive by its completeness and by the coherence of its page versions. Completeness measures the ability of the archive to contain

* This research is supported by the French National Research Agency ANR in the CARTEC Project (ANR-07-MDCO-016).

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 394–409, 2011.

the largest amount of useful versions. Coherence measures how much the archive reflects the snapshot of web sites at different points in time. An ideal approach to preserve the quality of archives is to crawl all pages of a site at the same time at every modification or to prevent pages content from changing during the crawl. Of course, this is practically infeasible because web sites are autonomous and thus out of control. In fact, it is impossible to maintain a complete archive of the whole web site (*i.e.* containing all the versions of all the pages) because web sites are evolving over time and allocated resources are usually limited (such as bandwidth, storage space, site politeness rules). Also, crawling a large web site may span hours and even days. This increases the risk of page changes during the crawl that leads to incoherence between archived pages. Page versions (of a same site) are considered incoherent, if they have never existed together at any point in time in the real web history.

Though it is impossible to preserve a perfect quality of web archives, this quality can be improved if web sites are crawled "at the right moment". Our work aims to adjust the crawling strategy so that the built archive will be as complete and as coherent as possible. Our ideas, for that, is (i) to improve the completeness of the archive by downloading the most *important* versions and (ii) to assure that coherent versions we obtain, are the most important ones. An important version is a version that has important change with respect to the last one archived of the same page. Hence, unimportant changes in the page (*e.g.* advertisements, decoration, etc.) can be ignored and useful information is captured by a single crawl, maximizing the use of resources. Up to now, most crawling strategies [5,13,10,16] do not consider the *importance of changes* that have occurred between versions. They consider the crawl useful even if the captured version is almost equal to the previous one. Moreover, they estimate the frequency of page changes based on the homogeneous Poisson model [8,6] with a constant change rate λ. This model is valid when the time granularity of changes is longer than one month as shown in [15] which is far from being the common case. For instance, our work is applied on a repository for the French National Audiovisual Institute (INA) which creates a legal deposit to preserve French radios and televisions web pages and related pages. Those web pages, such as on-line newspapers change very frequently (more than once a day). As the time granularity, for those pages, is much shorter than one month, the homogeneous Poisson model is not valid as demonstrated in [15].

We have the idea to use page change patterns. A pattern models the behavior of the importance of the changes over time, during for example a day. Based on patterns, the evolution of changes can be accurately predicted over periods of time and exploited to optimize crawlers. In previous work [3], we have monitored French TV channels pages (*France Télévision*) over a period of one month. Each page was hourly crawled every day. Then, we have discovered patterns by using a statistical summarization technique. Based on these patterns, we show, in this paper, how the strategy of crawlers can be adjusted to improve quality of archives. As far as we know, the concepts of changes importance and patterns had never been exploited to optimize the quality of archives. Related crawl policies

that have considered the importance/relevance of pages are mostly based on the PageRank (also similarity to keywords of queries) but the importance of changes between versions have been ignored so far. Moreover, this work is the first to address both completeness and coherence, at same time, in the context of web archiving. The main contributions of this paper can be summarized as follows:

- A description of our archiving model based on two concepts: changes importance and page changes pattern.
- A definition of two quality measures (completeness and coherence) to assess the quality of the archive. These measures consider the importance of archived versions.
- A novel crawl strategy based on patterns that uses the importance of changes to improve both the completeness and the coherence of web archives.
- An implementation of our strategy and experimental results that demonstrate its effectiveness.

This paper is structured as follows. In Section 2, related works are presented and discussed. Section 3 introduces the different concepts used in the paper. Section 4 describes our web archiving model. Section 5 defines two measures, completeness and coherence that assess the quality of archives. Section 6 describes our pattern-based strategy using the importance of changes. Section 7 discusses experimental results. Section 8 concludes.

2 Related Works

In recent years, several projects have addressed issues involved by web archiving. An overview of these main issues is given by Masanès [12]. Many studies are closely related to our work, in the sense that they aim at optimizing crawlers. Brewington and Cybenko [4] estimate how often web sites must be re-indexed based on the analysis of page changes for more fresher indexes. Pandey and Olston [13] propose a recrawl scheduling strategy based on information longevity to improve the freshness of web pages. In [8], Cho et al. estimate the frequency of page changes based on the Poisson process. In other studies [6,7], they propose efficient policies to improve the freshness of web pages. In [9], they propose a crawl strategy to download the most important pages first based on different metrics (*e.g* similarity between pages and queries, rank of a page, etc.). The research of Castillo et al. [5] goes in same direction. They propose a crawl strategy that retrieves the best ranked pages. Most designed strategies that have considered the importance of pages are based on the PageRank (also similarity to keywords of queries) but do not take into account the importance of changes between versions. Moreover, existing crawl strategies are mostly based on change rate estimated by the Poisson Process. As already mentioned in [15], researchers demonstrate that the Poisson process is not valid for pages changing several times per day as it is the case in our context. Similarly to our work, Adar et al. [1] propose models and analysis to characterize the amount of change on the web at finer grain (frequent updates per day). However, they do not propose a

method to estimate the importance of (structural and content) changes detected between pages versions. According to them, their change analysis can be used to refine crawler policies but, no effective strategy has been proposed.

Recent studies address the issue of improving the quality of archives. Spaniol et al. [16] propose a crawling strategy in order to optimize the coherence of web sites captures. In [17], they present visualization strategies to help the archivist at understanding the nature of coherence defects. In another study [10], they define two quality measures (blur and sharp) to assess the quality of the archive and propose a framework, coined SHARC, to optimize site-capturing policies. The two approaches proposed in [10,16] to improve the coherence and the sharpness of the archive are based on multiple revisits of web pages. However, in our work, we assume that web crawlers have limited resources which prevent from revisiting a page too often. Also, the importance of changes between page versions has been ignored.

Our work is also related to pattern mining area. Patterns are widely introduced and implemented for different applications such as trajectories of objects, weather, DNA sequences, stock market analysis, etc. They were exploited to detect anomalies, to predict data behavior (or trend), or more generally, to simplify data processing, etc. It is impossible to give here a complete coverage on this topic but interested readers can refer to [11] for example. A large coverage of pattern mining approaches is given. To the best of our knowledge, patterns have never been used to improve web archiving. In [3], we presented, through a case study, steps and methods to discover patterns from French TV channels pages. Here, we investigate how these patterns can be used to improve completeness and coherence of web archives based on the importance of changes.

3 Concepts

In order to better understand the next sections, we introduce here the different concepts that we use in this paper.

- **Quality of an archive.** We evaluate the quality of the archive through the two following measures.
  - ***Completeness.*** It measures the ability of the archive to contain the largest amount of useful page versions. This quality measure is relevant because it is very frequent, while navigating through the archive, that users cannot reach some web pages. Those missed pages had not been downloaded (at right moment) before they disappear from the web.
  - ***Coherence.*** It measures the ability of the archive to reflect the states (or snapshots) of a web site at different points in time. Indeed, when users navigate through the archive, they may want to browse (part of) sites instead of individual pages. Coherence ensures that if users reach a page version of a site, they can also reach other pages versions of the same site corresponding to the same point in time.

  In the rest of the paper, we use the term quality to express both completeness and coherence of the archive.

- **Importance of a version**

An important version is a version of an important page that has significant changes compared with the last version archived of the same page. Therefore, the importance of a version depends on:

1. the importance of the corresponding page (*e.g.* PageRank, similarity to keywords of a query, etc.)
2. the importance of the changes that have occurred on the page since the last version archived.

Changes between two page versions are detected by the Vi-DIFF algorithm [14]. First, Vi-DIFF extends a visual segmentation algorithm to partition the web page into multiple blocks. Blocks simulate how a user understands the page layout structure based on his visual perception. Then, Vi-DIFF detects *structural* changes (*i.e.* an insert, a move, etc. at level of blocks composing the page) and *content* changes (*i.e.* a delete, an update, etc. at level of texts, hyperlinks and images inside blocks). The importance of changes between two versions is estimated by the following function E.

$$E = \sum_{i=1}^{N_{Bk}} ImpBk_i * \frac{1}{N_{Op}} \sum_{j=1}^{N_{Op}} ImpOp_j * PerCh_{i,j}$$

where

- *ImpBk* is the importance of each block composing the page. The importance of a block in the page depends on its location, area size, content, etc.
- *ImpOp* is the importance of changes operations (insertion, deletion, etc.) detected between the two versions. For instance, delete operation can be considered less important than an insert or an update.
- *PerCh* is the percentage of changes (insert, delete, etc.) occurred on each block with respect to the total number of block's elements.
- N_{Op}, N_{Bk} are respectively the number of change operations and the number of blocks in the page.
- $\sum_{i=1}^{N_{Bk}} ImpBk_i = 1$

The estimator E returns a normalized value between 0 and 1. This value assesses the importance of changes between two page versions. The larger the number of significant changes occurred inside important blocks is, the higher the estimated importance of changes. For more details about the algorithm Vi-DIFF used to detect changes between two versions of pages, please refer to [14]. The estimator of the importance of changes is detailed in [2].

Page Changes Pattern

Periods T	Workdays ω_k	Saturday ω_k	Sunday ω_k
[0:00-6:00]	0,2	0,1	0,2
[6:00-12:00]	0,4		
[12:00-18:00]	0,6	0,4	0,35
[18:00-24:00]	0,1	0,2	0,13

Fig. 1. Pattern Example

- **Page changes pattern**
  A pattern models the behavior of page's changes over periods of time, during for example a day. It is periodic and may depend on the day of the week and of the hour within a day. An example of pattern is shown in Figure 1. It defines the importance of changes over different periods of the day. Separate patterns can be defined for weekends.

4 Web Archive Model

In our web archive model, all the web pages are repeatedly captured individually. The crawler typically works at the granularity of a page and not at the granularity of a web site. It selects the most important pages to be refreshed from a large collection of URLs under a resource constraint, *e.g.* one page crawled per second. Pages that change very frequently with a significant modification, are visited more often. Our web archive A_S is defined as a set of archived sites A_{S_i}. A_{S_i} is a set of versions of pages downloaded from a site S_i. An interval of observation [o_s,o_e] is defined for the archive where o_s is the starting time and o_e is the ending time as shown in the Figure 2. The archive A_{S_i} is accessed by a user query $Q(t_q, A_{S_i})$ that browses the closest available page versions of a site S_i at a given time query t_q. Our work aims at improving the quality of versions returned to user for any time t_q.

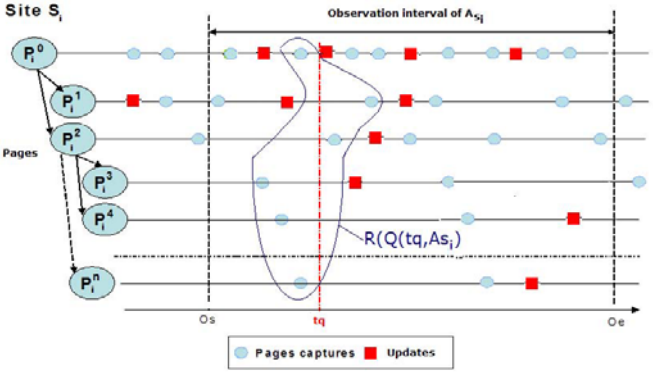


Fig. 2. Web Archive Model

4.1 Assumptions

In the following, we assume that the pages to be crawled change over time independently from each other. Patterns are considered known for any page and have already been discovered by using the approach proposed in [3]. We assume that the web crawler has limited resources for capturing new versions of pages. We model the resource constraint by assuming that the crawler can download a

total of M pages in each period T. We assume that all snapshots (or states) of web sites to be crawled are coherent at each instant. This means that each site, at any time point, do not present conflicting information such as broken links, error of posting pictures, etc. This type of incoherence is out of control and is ignored in this work.

4.2 Notation and Definitions

We assume that S={S_1,S_2,...,S_κ} is the list of sites to be crawled. Each site S_i consists of N_i pages {P_i^1,P_i^2,...,$P_i^{N_i}$}. Each page P_i^j has a pattern Patt(P_i^j). In addition, we assume that the importance of the page P_i^j is $\omega(P_i^j)$. The importance of a change occurred on the page P_i^j at instant t is denoted by $\omega_i^j[t]$.

Definition 4.21. *Pattern*

A pattern of a page P_i^j with an interval length l is a nonempty sequence $Patt(P_i^j) = \{(\omega_1,T_1); ...; (\omega_k,T_k);...; (\omega_{N_T},T_{N_T})\}$, where N_T is the total number of periods in the pattern and ω_k is the average of the importance of changes estimated in the period T_k. The sum of the time periods, $\sum_{k=1}^{N_T} T_k$, is equal to l.

We note $P_i^j[t]$ the version of the page P_i^j captured at time t in the archive A_{S_i}. We note $\tilde{P}_i^j[t]$ the version of the page P_i^j created (by a change) on the real web site S_i at time t. As shown in the Figure 3, the page P_i^j has one capture at time t_1 that corresponds to the archived version $P_i^j[t_1]$. The two versions $\tilde{P}_i^j[\tau_1]$ and $\tilde{P}_i^j[\tau_2]$ created on the web site S_i correspond to the two changes occurred on the page P_i^j at time τ_1, τ_2.

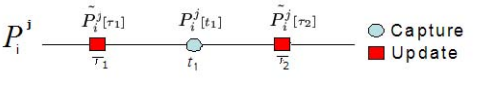


Fig. 3. Example of page versions

Definition 4.22. *Archive*

Let κ be the total number of sites, the archive A_S is the set of archived sites A_{S_i}, $S_i \in S$.

$$A_S = \bigcup_{i=1}^{\kappa} A_{S_i}$$

The archive A_{S_i} of a site S_i is defined by the set of page versions $P_i^j[t]$ captured from the site S_i during the interval $[o_s, o_e]$.

$$A_{S_i} = \{P_i^j[t], 1 \leq j \leq N_i | P_i^j \in S_i \ \wedge t \in [o_s, o_e]\}$$

Definition 4.23. *User Query*

The user query $Q(t_q, A_{S_i})$ asks for the closest snapshot of the site S_i to the query time t_q.

Definition 4.24. *Query Result*

The result $R(Q(t_q, A_{S_i}))$ *of the user query* $Q(t_q, A_{S_i})$ *is the set of the* N_i *versions* $P_i^j[t]$ *(one for each page of* S_i*) which are the closest to the time* t_q *as shown in Figure 2.*

$$R(Q(t_q, A_{S_i})) = \{P_i^j[t] \in A_{S_i} | \neg\exists P_i^j[t'] \in A_{S_i} : |t^{'} - t_q| < |t - t_q|\}; j = \{1, .., N_i\}$$

Definition 4.25. *Version Importance*

Let $P_i^j[t]$ *be the version of the page* P_i^j *that has been captured at time t after the change occurred at time t'. The importance* $\omega(P_i^j[t])$ *of the version* $P_i^j[t]$ *is the multiplication of the importance of its corresponding change* $\omega_i^j[t']$ *by the importance of the page* $\omega(P_i^j)$*.*

$$\omega(P_i^j[t]) = \omega_i^j[t'] * \omega(P_i^j)$$

where t' is the time of the last change of $P_i^j[t]$ *preceding t.*

5 Quality Measures

We define, here, two quality measures completeness and coherence that take into account the importance of versions.

5.1 Completeness

The completeness of archives measures the proportion of the importance of changes that have been captured with respect to the total amount of the importance of changes that occurred on web sites.

Definition 5.11. *Complete Archive*

An archive is complete, if it contains all the versions of pages $\tilde{P_i^j}[t]$ *that appeared on all sites composing the archive.*

$$\forall \tilde{P_i^j}[t], \exists P_i^j[t'] \in A_{S_i}, t' \geq t : P_i^j[t'] = \tilde{P_i^j}[t]$$

Definition 5.12. *Archived Page Completeness*

The completeness of an archived page P_i^j *is the sum of the weights of versions that have been captured, divided by the total weight of versions (created by changes) that appear on the real web site . Let m be respectively the number of versions* $P_i^j[t_k]$ *captured at time* t_k *and p be the number of versions* $\tilde{P_i^j}[\tau_k]$ *created on the site at time* τ_k*, the completeness of archived page* P_i^j *is*

$$Completeness(P_i^j) = \frac{\sum_{k=1}^{m} \omega(P_i^j[t_k])}{\sum_{k=1}^{p} \omega(\tilde{P_i^j}[\tau_k])}$$

where
- The weight of the version $\omega(P_i^j[t_k])$ *is equal to the last change importance* $\omega_i^j[t']$
- The weight of the version $\omega(\tilde{P_i^j}[\tau_k])$ *is equal to the last change importance* $\omega_i^j[\tau']$
- $\omega_i^j[t']$ *and* $\omega_i^j[\tau']$ *denote the importance of the changes that occurred respectively at t' and* τ' *just before the capture of the versions* $P_i^j[t_k]$ *and* $\tilde{P_i^j}[\tau_k]$.

Definition 5.13. *Archived Site Completeness*

The completeness of archived site A_{S_i} *is the sum of the completeness of the* N_i *pages of* S_i *(weighted by their importance) divided by the overall pages importance.*

$$Completeness(A_{S_i}) = \frac{\sum_{j=1}^{N_i} Completeness(P_i^j) * \omega(P_i^j)}{\sum_{j=1}^{N_i} \omega(P_i^j)}$$

Definition 5.14. *Archive Completeness*

The overall completeness of the archive A_S *is the average completeness of all archived sites.*

$$Completeness(A_S) = \frac{\sum_{i=1}^{\kappa} Completeness(A_{S_i})}{\kappa}$$

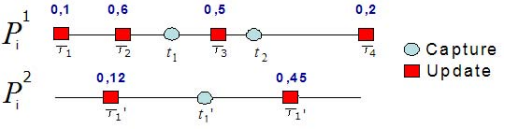


Fig. 4. Site Completeness Example

Example 5.11. *Archived Site Completeness.*

We consider a site S_i *consisting of two pages* P_i^1 *and* P_i^2 *as shown in Figure 4. We assume that* $P_i^1[t_1]$ *and* $P_i^1[t_2]$ *are the two versions of the page* P_i^1 *captured respectively at time* t_1 *and* t_2. *We assume that the importance of the four changes that have occurred on the page* P_i^1 *at time* τ_1, τ_2, τ_3 *and* τ_4 *are respectively 0.1, 0.6, 0.5, 0.2. For the page* P_i^2, *there is one capture* $P_i^2[t_1']$ *at* t_1'. *The importance of the two changes occurred on the page* P_i^2 *at* τ_1' *and* τ_2' *are respectively 0.12 and 0.45.*

Note that in this example (and also in example 5.21), we assume that the importance of each page is equal to 1 ($\omega(P_i^j) = 1, \forall i, j$).
The completeness of P_i^1 and P_i^2 is

$$Completeness(P_i^1) = \frac{\omega(P_i^{1[t_1]}) + \omega(P_i^{1[t_2]})}{\sum_{k=1}^{4} \omega(P_i^{1[\tau_k]})} = \frac{0.6+0.5}{0.1+0.6+0.5+0.2} = 0.78$$

$$Completeness(P_i^2) = \frac{\omega(P_i^{2[t_1']})}{\omega(P_i^{2[\tau_1']}) + \omega(P_i^{2[\tau_2']})} = \frac{0.12}{0.12+0.45} = 0.21$$

The overall completeness of archived site A_{S_i} is

$$Completeness(A_{S_i}) = \frac{Completeness(P_i^1) * \omega(P_i^1) + Completeness(P_i^2) * \omega(P_i^2)}{\omega(P_i^1) + \omega(P_i^2)}$$

$$Completeness(A_{S_i}) = \frac{0.78*1+0.21*1}{2} = 0.49$$

5.2 Coherence

A collection of archived pages versions is considered *coherent*, if it reflects the state (or the snapshot) of the web site at, at least, one point in time. Our definition of coherence is inspired by the approach of Spaniol and al. In [16], they introduce two measures to quantify the coherence of a site crawl. Their measures count the expected number of occurring incoherences during a complete crawl of a site. They are based on either (i) the last modified stamp or (ii) on a virtual time stamp obtained by revisiting each page. We do not use these measures because the last modified stamp is not always trustful in real life crawls. The virtual time stamp assumes that, during an on-line crawl, each page must be revisited twice in a short time. As we assume that the crawler has limited resources, we do not use virtual time stamps. We propose a new measure, inspired by Spaniol's definition [16], that considers the importance of changes to quantify the coherence of the query result.

Definition 5.21. *Coherent Versions*

The N_i versions of $R(Q(t_q, A_{S_i}))$ are coherent, if there is a time point (or an interval) so that it exists a non-empty intersection among the invariance interval $[\mu_j, \mu_{j^*}]$ of all versions.

$$\forall P_i^j[t] \in R(Q(t_q, A_{S_i})), \exists t_{coherence} : t_{coherence} \in \bigcap_{j=1}^{N_i} [\mu_j, \mu_{j^*}] \neq \varnothing \qquad (1)$$

where μ_j and μ_j^* are respectively the previous and the next changes following the capture of the version $P_i^j[t]$.

As shown in Figure 5 at the left, the three versions $P_i^1[t_1]$, $P_i^2[t_2]$ and $P_i^3[t_3]$ of $R(Q(t_q, A_{S_i}))$ are coherent because there is an interval $t_{coherence}$ that satisfies the coherence constraint (1). However the three page versions at the right are not coherent because there is no point in time satisfying the coherence constraint (1).

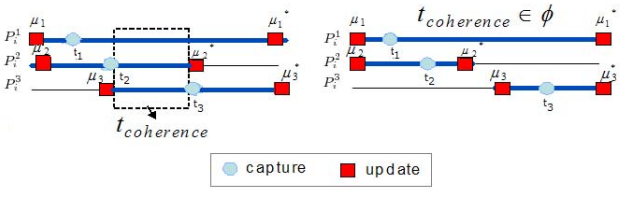


Fig. 5. Coherence Example [16]

Definition 5.22. *Query Result Coherence*

The coherence of the query result $R(Q(t_q, A_{S_i}))$ is the weight of the largest number of coherent versions divided by the total weight of the N_i versions of $R(Q(t_q, A_{S_i}))$. We assume that $\{P_i^1[t_1],..., P_i^\rho[t_\rho]\} \in R(Q(t_q, A_{S_i}))$ are the ρ coherent versions, i.e satisfying the constraint (1). We assume that ρ is the

largest number of coherent versions composing $R(Q(t_q, A_{S_i}))$.
The coherence of $R(Q(t_q, A_{S_i}))$ *is*

$$Coherence(R(Q(t_q, A_{S_i})) = \frac{\sum_{k=1}^{\rho} \omega(P_i^k[t_k])}{\sum_{k=1}^{N_i} \omega(P_i^k[t_k])}$$

where $\omega(P_i^k[t_k])$ *is the importance of the version* $P_i^k[t_k]$.

Definition 5.23. *Site Archive Coherence*

The overall coherence of archived site A_{S_i} *can be estimated through the average coherence of* $R(Q(t_q, A_{S_i}))$ *obtained for different time query* t_q.

$$Coherence(A_{S_i}) = \frac{\sum_1^{n_Q} Coherence(R(Q(t_q, A_{S_i})))}{n_Q}$$

where n_Q *is the number of queries that have accessed the archive* A_{S_i} *in the observation interval* $[o_s, o_e]$.

Definition 5.24. *Archive Coherence*

The overall coherence of the archive A_S *is the average coherence of the* κ *archived sites.*

$$Coherence(A_S) = \frac{\sum_{i=1}^{\kappa} Coherence(A_{S_i})}{\kappa}$$

Example 5.21. *Query Result Coherence*

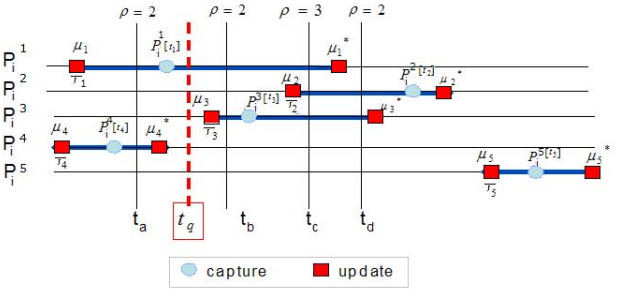


Fig. 6. $R(Q(t_q, A_S))$ Example

We assume that $P_i^1[t_1]$, $P_i^2[t_2]$, $P_i^3[t_3]$, $P_i^4[t_4]$, $P_i^5[t_5]$ *are the five versions of* $R(Q(t_q, A_{S_i}))$ *which are the closest to the time query* t_q. *We assume that* $\omega_i^1[\tau_1] = 0.2$, $\omega_i^2[\tau_2] = 0.54$, $\omega_i^3[\tau_3] = 0.34$, $\omega_i^4[\tau_4] = 0.22$ *and* $\omega_i^5[\tau_5] = 0.6$ *are the importance of changes* $\mu_1, ..., \mu_5$ *occurred respectively on the pages* P_i^1, ...,P_i^5 *at instants* $\tau_1, ..., \tau_5$.

The value ρ *of the largest number of coherent versions is equal to 3 because there are three coherent versions* $P_i^1[t_1]$, $P_i^2[t_2]$ *and* $P_i^3[t_3]$ *satisfying the constraints (1) : it exists a time* $t_c \in [\mu_1, \mu_1^*] \bigcap [\mu_2, \mu_2^*] \bigcap [\mu_3, \mu_3^*]$.
Then, the coherence of $R(Q(t_q, A_{S_i}))$ *is*

$$Coherence(R(Q(t_q, A_{S_i}))) = \frac{\omega(P_i^1[t_1]) + \omega(P_i^2[t_2]) + \omega(P_i^3[t_3])}{\sum_{k=1}^{5} \omega(P_i^k[t_k])}$$
$$= \frac{0.2 * 1 + 0.54 * 1 + 0.34 * 1}{0,2 * 1 + 0.54 * 1 + 0.34 * 1 + 0.22 * 1 + 0.6 * 1} = 0.56$$

6 Pattern-Based Web Crawling

To improve the quality of archives, we propose a crawling strategy directly driven by the patterns defined in Section 4.2. Our goal is to schedule page crawls in such a way that it improves completeness and coherence of the archive. The crawler can download a total of M pages at each period T_k. To simplify notations, we assume, in the remainder of the paper, that $P_1, P_2, ..., P_n$ is the list of all pages to be crawled. By using patterns, pages are scheduled based on their urgency (or priority). Each page is assigned an urgency value *U(P$_i$,t)* proportional to both the expected changes importance ω_k (defined by pattern at period T_k) and to the importance of the page $\omega(P_i)$. Also, the urgency of pages changes with the time. It depends on the time of the last refresh and on the current time. The urgency *U(P$_i$,t)* of the page P_i at time t is
$U(P_i, t) = \omega(P_i) * \omega_k * (t - t_{lastRefresh})$
where

- Patt(P$_i$) = {(ω_1,T_1); ...; (ω_k,T_k);...; (ω_{N_T},T_{N_T})},
- t is the current time (t $\in T_k$),

Algorithm 1. Pattern-based Crawler

Input:
P$_1$, P$_2$,..., P$_n$ - list of pages
Patt(P$_1$), Patt(P$_2$),..., Patt(P$_n$) - patterns of pages
Begin
1. **for** each period T_k **do**
2. crawlListPages⟵newList()
3. **for** each page P$_i$, i=1...,n **do**
4. compute $U(P_i, t) = \omega(P_i) * \omega_k * (t_i - t_{ilastRefresh})$
5. crawlListPages.add(P$_i$,$U(P_i, t)$) /* in descending order of urgency */
6. **end for**
7. **for** i=1...,M **do**
8. P$_i$⟵crawlListPages.selectPage(i)
9. currentVersion⟵downloadPage(P$_i$)
10. lastVersion⟵getLastVersion(P$_i$)
11. delta⟵detectChanges(currentVersion,lastVersion)
12. ω⟵EstimateChangesImportance(delta)
13. Update(Patt(*page*),ω,T_k)
14. $t_{ilastRefresh} \longleftarrow t_i$
15. **end for**
16. **end for**
End

- ω_k is the average of change importance defined by Patt(P_i) in period T_k,
- $\omega(P_i)$ is the importance of the page,
- $t_{lastRefresh}$ is the last time of refreshing the page P_i.

At each period T_k, only the M-top pages with the highest current priority are captured. The M selected pages are downloaded in descending order of their urgency $U(P_i, t)$. Afterwards, each captured page version is compared with its predecessor to detect changes based on the Vi-DIFF algorithm (*cf.* Section 3). Then, the importance of changes can be estimated by the function E (*cf.* Section 3) and exploited to update patterns. Patterns need to be updated periodically to always reflect the current changes of web pages. Thus, the average change importance ω_k defined by patterns in period T_k is periodically updated during an on-line crawl. Also, the importance of page (*e.g.* PageRank) is regularly reevaluated over time to reflect the real web. The pseudo code of the pattern-based strategy is depicted by Algorithm 1.

7 Experimental Evaluation

In this section, we evaluate the effectiveness of our crawling approach by comparing it with existing strategies. In particular, we compare the total completeness and coherence (*cf.* Section 5) obtained by each policy. As it is impossible to obtain exactly all the versions that appear on real web sites, we have simulated the change importance of web pages based on real patterns discovered from "France Télévisions" channels pages [3]. Experiments written in Java were conducted on PC running Linux over a 3.20 GHz Intel Pentium 4 processor with 1.0 GB of RAM. At the begin of each experiment, each page is described by a real pattern. The updates rate and the changes importance of each page is generated according to defined patterns. In addition, the following parameters are set: the number of pages per site (one thousand pages), the duration of simulation, the number of periods in patterns (24 periods), the number of allocated resources (*i.e.* the maximum number of pages that can be captured per each time period). Equal resources are assigned to different crawler strategies to evaluate them under the same constraints.

We start by describing related strategies considered in this work: ***Relevance*** [9] downloads the most important pages (*i.e* based on PageRank) first, in a fixed order. ***Frequency*** [7] selects pages to be archived according to their frequency of changes estimated by the Poisson model [8]. Hot pages that change too often are penalized to maximize the freshness of pages. ***Coherence*** [16] works at the granularity of a site and downloads firstly the less "risky" pages (*i.e.* with lowest probability to cause incoherences). Then, it continues by capturing the remaining pages that had been skipped. ***SHARC*** [10] repeatedly downloads the entire sites and ensures that the most changing page are downloaded as close as possible to the middle of the capture interval. ***Importance*** is our first strategy which selects pages based on their urgency (*cf.* Section 6). The parameter of changes importance ω_k is a fixed weight (average) and does not depend on time periods

T_k. This strategy consider the importance of changes without using patterns. ***Pattern*** is our second strategy which depends only on patterns without considering the importance of changes. It downloads pages according to their urgency based on changes rate instead on changes importance. ***Importance-Pattern*** is our third strategy which downloads pages based on their urgency (*cf.* Algorithm 1). It combines the two concepts importance of changes and patterns.

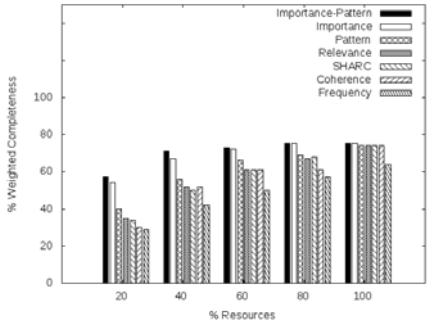


Fig. 7. Weighted Completeness

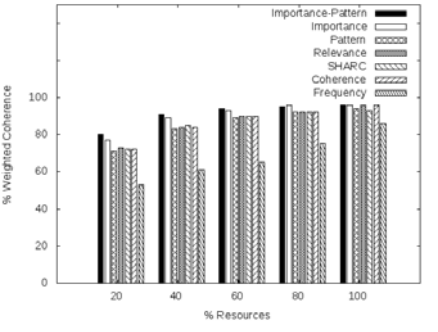


Fig. 8. Weighted Coherence

We evaluated the completeness (*cf.* Section 5.1) obtained by the different crawl strategies as shown in the Figure 7. The horizontal axis shows the percentage of allocated resources M=[20%-100%]. The vertical axis shows the weighted completeness that have been captured by each policy. As we can see, the completeness increases with the number of allocated resources. Obviously, when more pages are captured at each period, better completeness is achieved. We notice also that it is impossible to achieve 100% of completeness even if 100% of resources are allocated. There are always some missed versions. From the figure, it is clear that our strategy *Importance-Patterns* performs better than its competitors *Relevance*, *SHARC*, *Coherence* and *Frequency*. It improves the completeness of the archive by around 20% in case of limited resources. Also, it performs 5% better than *Importance* strategy which does not use patterns. This gain is rather low in average, but we note that it can reach more than 10 % in case the patterns of web pages are significantly different one from the others.

We evaluated also the coherence of the archive based on the measure defined in Section 5.2. Figure 8 shows the percentage of coherence weight achieved by different crawl strategies with respect to the number of allocated resources. As we can see, our *Importance-Patterns* strategy achieves the highest coherence weight. It performs around 10% better than its competitors *SHARC*, *Coherence* and *Relevance*. *Frequency* strategy achieves the lowest coherence weight. To sum up, these experiments demonstrate that *Importance-Patterns* improves both the completeness and the coherence of archives by respectively 20 % and 10 %.

8 Conclusion and Future Work

Preserving the quality of web archives is a crucial issue addressed by archivists nowadays. Here, we point out the issue of efficiently crawling web pages in order to improve the quality of archives. We defined two metrics to measure the quality of the archive. Completeness measures the ability of the archive to contain the largest amount of useful information. Coherence measures how much the archive reflect the snapshot of web sites at different points in time. As far as we know, this work is the first to formalize and to address both issues (coherence and completeness) at the same time. Our challenge is to adjust the crawl strategy to make the archive as complete and as coherent as possible. We propose a pattern-based strategy which use the importance of changes to improve the quality of the archive. To the best of our knowledge, the concepts of the importance of changes and patterns have never been used to improve the quality of archives. Most related strategies download with priority the most frequently changing pages (and/or the most important ones based on PageRank) but do not consider the importance of changes occurred between page versions. Conducted experiments based on real patterns confirm that our pattern-based strategy outperforms its competitors. Results show that it is able to improve the completeness of the archive by around 20% and the coherence by around 10 % in case of limited resources. This improvement of the archive quality can be furthermore better, if patterns of web pages are significantly different one from the others.

We are currently pursing our study to run our pattern-based strategy over a large number of web pages collected by the National Audiovisual Institute (INA). We are also studying how patterns can be exploited to decide when page versions should be indexed or stored. Hence, archive systems will avoid wasting time and space for indexing/storing unimportant pages versions. Further study must be done to learn how we can create a collection of common patterns for pages with similar behavior of changes. An other on-going work is to find an efficient method to maintain patterns up-to-date during an on-line crawl.

References

1. Adar, E., Teevan, J., Dumais, S.T., Elsas, J.L.: The web changes everything: understanding the dynamics of web content. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining, Barcelona, Spain (2009)
2. Ben Saad, M., Gançarski, S.: Using visual pages analysis for optimizing web archiving. In: EDBT/ICDT PhD Workshops, Lausanne, Switzerland (2010)
3. Ben Saad, M., Gançarski, S.: Archiving the Web using Page Changes Pattern: A Case Study. In: ACM/IEEE Joint Conference on Digital Libraries (JCDL 2011), Ottawa, Canada (2011)
4. Brewington, B.E., Cybenko, G.: Keeping up with the changing web. Computer 33(5) (2000)
5. Castillo, C., Marin, M., Rodriguez, A., Baeza-Yates, R.: Scheduling algorithms for web crawling. In: LA-WEBMEDIA 2004: Proceedings of the WebMedia (2004)

6. Cho, J., Garcia-Molina, H.: The Evolution of the Web and Implications for an Incremental Crawler. In: VLDB 2000: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 200–209. San Francisco, CA, USA (2000)
7. Cho, J., Garcia-Molina, H.: Effective page refresh policies for web crawlers. ACM Trans. Database Syst. 28(4), 390–426 (2003)
8. Cho, J., Garcia-molina, H.: Estimating frequency of change. ACM Transactions on Internet Technology 3, 256–290 (2003)
9. Cho, J., Garcia-molina, H., Page, L.: Efficient crawling through url ordering. In: Computer Networks and ISDN Systems, pp. 161–172 (1998)
10. Denev, D., Mazeika, A., Spaniol, M., Weikum, G.: Sharc: framework for quality-conscious web archiving. Proc. VLDB Endow. 2(1), 586–597 (2009)
11. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. In: Data Mining and Knowledge Discovery, vol. 15 (2007)
12. Masanès, J.: Web Archiving. Springer, New York (2006)
13. Olston, C., Pandey, S.: Recrawl scheduling based on information longevity. In: Proceeding of the 17th International Conference on World Wide Web (2008)
14. Pehlivan, Z., Ben-Saad, M., Gançarski, S.: Vi-DIFF: Understanding web pages changes. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6261, pp. 1–15. Springer, Heidelberg (2010)
15. Sia, K.C., Cho, J., Cho, H.-K.: Efficient monitoring algorithm for fast news alerts. IEEE Transactions on Knowledge and Data Engineering 19, 950–961 (2007)
16. Spaniol, M., Denev, D., Mazeika, A., Weikum, G., Senellart, P.: Data quality in web archiving. In: WICOW 2009: Proceedings of the 3rd Workshop on Information Credibility on the Web, pp. 19–26 (2009)
17. Spaniol, M., Mazeika, A., Denev, D., Weikum, G.: "catch me if you can": Visual analysis of coherence defects in web archiving. In: 9th International Web Archiving Workshop (IWAW 2009), Corfu, Greece, pp. 27–37 (2009)

Alternative Query Generation for XML Keyword Search and Its Optimization

Tetsutaro Motomura, Toshiyuki Shimizu, and Masatoshi Yoshikawa

Graduate School of Informatics, Kyoto University
motomura@db.soc.i.kyoto-u.ac.jp, {tshimizu,yoshikawa}@i.kyoto-u.ac.jp

Abstract. Much work has been done in XML keyword search since users can obtain various information from XML databases without specific knowledge of the database schema and/or the knowledge about the query languages. Moreover, certain researches have suggested methods of returning some information that would help users understand search results. In this paper, we define alternative queries, which can be considered as different aspects of XML keyword search results. In XML keyword search, a keyword may match an unexpected text value or element name, then incorrect results that do not correspond to the users' search intentions may be retrieved. When we generate alternative queries, it does not seem useful to generate alternative queries for all the results since they include several results retrieved by several interpretations. Thus, we propose a method of generating alternative queries from results classified by interpretations. We also propose a stack-based algorithm for generating alternative queries. Finally, the experimental results reveal that our proposal generates alternative queries efficiently.

1 Introduction

XML has recently become a widely accepted standard for semi-structured documents and is used as a data description framework. It is advantageous to support users when they retrieve useful information from large amounts of XML data. Users can post queries and obtain various information from XML databases if user-friendly ways such as keyword search in Web search are supported.

While numerous studies on XML keyword search [1–3] have been done to efficiently retrieve results from large amounts of XML data, supporting methods to enhance the result quality were not explored so far. It is generally difficult for uses to obtain search results by matching their search intentions in a single trial. When this occurs, they have to refine their queries and conduct searches many times unless they obtain the results that they intended to search for. Thus, it is advantageous to not only support efficient search methods but also support methods that pose some useful information that can refine users' queries and/or help them to understand the results more fully in XML keyword search.

The understanding of XML search results has been studied as the extraction of a summary of an XML subtree [4] and a set of differentiation features of an XML subtree [5]. Several different approaches have been proposed in the

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 410–424, 2011.

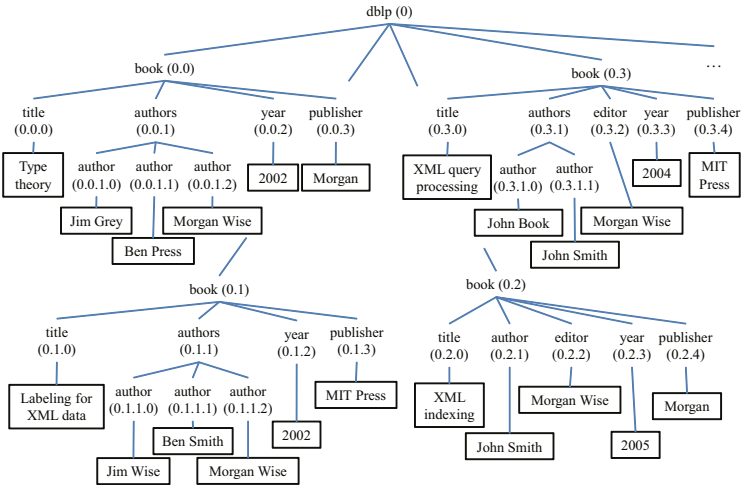


Fig. 1. Literature database represented as XML tree

area of relational database, i.e., database structure mining [6, 7], intentional query answering [8], and the discovery of frequent co-occurring terms [9]. Tran et al. [10] suggested a data-driven approach called *Query by Output (QBO)* where their goal was to obtain a query Q' such that $Q(D)$ and $Q'(D)$ are instance-equivalent when given the output of original query Q on database D.

Instance-Equivalent Queries (IEQs) can be regarded as different aspects of search results. Thus, providing IEQs to users is considered useful to help them understand more fully. For example, the results queried by input keywords "Smith editor Morgan" and the results queried by keywords "XML John Smith" over the literature database represented as a XML tree in Figure 1 are equal when the Exclusive Lowest Common Ancestor (ELCA) [1] approach is adopted, where both results are subtrees rooted at 0.2 and 0.3. Here, ELCA computes nodes that contain all query keywords but has no nodes that contain all the keywords on the path between the ELCA nodes and the nodes that input keywords appears. When this occurs, a user infers that there is a close relationship between John and Morgan and XML and Smith since two books whose editor is Morgan authored by John have different aspects indicating the books include a specific keyword "XML" and an author name "Smith". Since ELCA semantics is considered to be able to provide more reasonable results, we adopted it as a typical concept in search results.

There is also a known problem in XML keyword search referred to by Bao et al. [11] where some input keywords match nodes users do not expect them to match, and this causes results to be retrieved that do not match the users' search intentions. Two factors are involved in matching unexpected nodes: (1) ambiguity where a keyword can appear both as an XML element name and as the text values of some other nodes, and (2) ambiguity where a keyword

can appear as the text values of different element nodes and carry different meanings. For example, ambiguity (1) in Figure 1 corresponds to where the keyword "book" appears as both an element name of XML and text values, and ambiguity (2) corresponds to where the keyword "Morgan" appears as the text value of the author, editor, and publisher node. While existing approaches calculate XML subtrees without giving consideration to these two problems with ambiguity, this is not beneficial in XML keyword search. Since there are generally several different interpretations when a search engine interprets an input keyword set, there are many multiple results obtained by many interpretations. It is not worthwhile generating alternative queries for whole XML keyword search results as these contain those obtained different interpretations. Thus, we believe that we should classify the raw search results into clusters by interpretations. More concretely, we make four main contributions that can be summarized as follows:

1. We defined the problem of generating alternative queries for XML keyword search taking into consideration ambiguities in keywords.
2. Moreover, we defined approximate alternative queries and two indicators to evaluate approximate alternative queries.
3. We implemented both naive and optimized methods to obtain approximate alternative queries. Our experiments revealed that the optimized method was more effective than the naive method.

The rest of this paper is organized as follows. We present related work in Section 2, and preliminary work on the data model, search concepts, and methods of obtaining nodes that satisfy these concepts in Section 3. Section 4 describes methods of generating alternative queries and approximate alternative queries. Experimental evaluations are discussed in Section 5 and we conclude the paper in Section 6.

2 Related Work

Extensive research has been done in the area of XML keyword search to find the smallest sub-structures in XML data. The Lowest Common Ancestor (LCA) [12] and the LCA family such as Smallest LCA (SLCA) [2], Exclusive LCA (ELCA) [1], and Meaningful LCA (MLCA) [3] have been studied. The works proposed in [13, 14] infer meaningful fragments returned from input keywords. The works mentioned above do not consider the two ambiguities where a keyword can appear both as an element name and as a text value of another node and a keyword can appear as the text value of different element nodes. The XReal system [11] addresses these ambiguities with an IR-style approach that infers the users' search intentions. However, it is difficult to determine what users' search intentions are unless they are uniquely identified. Thus, we consider it would be better to classify the results into interpretations and provide them to users.

Huang et al. [4] proposed XML snippets as summaries of XML subtrees to enable XML search results to be understood. Liu et al. [5] suggested Differentiation Feature Sets (DFSs), which are sets that reflect the most important

features in search results to help users to compare them. Database structure mining [6, 7] has been proposed in the area of understanding database search results where the goal has been to find the structural relationships in database tables. Intentional query answering [8] has been proposed to augment query results with additional information to help users understand the results. Tao et al. [9] proposed the discovery of frequent co-occurring terms to refine an input query with these terms. *Query by Output (QBO)* [10] is a data-driven approach to obtain *instance-equivalent queries (IEQs)* from results in the relational model. QBO only takes into account whether a keyword appears in an entity or not. However, it is difficult to apply QBO to discovery tasks of IEQs in XML keyword search since results are obtained with both information from keywords and structures. Thus, a new method that takes into account text values and the structure of XML data is needed when we discover IEQs in XML search.

Approximate alternative queries can be considered as obtaining approximate query tasks. Query expansion methods [15, 16] in the area of information retrieval have been proposed to obtain more relative information by relaxing given input keywords. However, these methods cannot be applied to generating approximate alternative queries for XML keyword search since they do not consider structure information. Relaxation approaches [17, 18] in the area of XML search using path queries have been proposed to obtain exact and approximate results by relaxing structural conditions. Our goal was to generate approximate queries by mining from XML subtrees with consideration given to the structure of information in the XML tree.

As query optimization is one of the most important areas in the classic relational model, extensive research has been done [19, 20] on it. However, it is important to pose diverse alternative queries in generating alternatives. Thus, our goal in this work was not to obtain optimized queries that performed better than the original queries but to generate diverse alternative queries.

3 Preliminaries

We model XML trees as ordered and labeled trees [1]. Each node v of an XML tree other than leaf nodes corresponds to an XML element node and is labeled with tag $\lambda(v)$. For the sake of simplicity, let us consider attribute nodes as element nodes. Each node v has a unique ID. Existing studies [1, 2] have used the *Dewey order* as the node ID. The Dewey order of a node includes the Dewey order of the parent node as a prefix. Since prior work [1] has shown that the Dewey order works well in a stack algorithm, we assign each node other than leaf nodes to a Dewey order. We represent the Dewey order of node v as $dewey(v)$. We also represent the subtree rooted at v as $st(v)$ or $st(dewey(v))$.

We utilize inverted lists where we can generate alternative queries. The inverted list of word k is generally a list of nodes directly containing k. *Dewey Inverted Lists (DILs)* [1] are inverted lists of XML trees labeled with a Dewey

[1] For simplicity, we assumed that a node in XML trees would have a different element name from others that appeared as ancestor or descendant nodes of the node.

order. We believe keywords can appear in several different element nodes. Moreover, we also consider where keywords can appear as element names. Thus, we create extended DILs such that each entry consists of: (1) the Dewey order and the element name of the node whose text value includes k, or (2) only the Dewey order of a node whose element name is k. We denote a pair of a keyword and its parent element name as a *tagged keyword*. As each entry s in inverted lists corresponds to a node, we call the entry node s.

Given k nodes $v_1, v_2, ..., v_k$ in XML tree T, the *Lowest Common Ancestor (LCA)* of the nodes $v_1, v_2, ..., v_k$ is a node that has v_1, v_2, ..., v_k as descendant nodes and its position is lowest in T. Let $lca(v_1, v_2..., v_k)$ be the function that obtains the LCA of v_1, v_2, ...Cv_k. Given node set $V = \{v_1, v_2, ..., v_n\}$, keyword set $K = \{k_1, k_2, ..., k_n\}$. *Exclusive Lowest Common Ancestor (ELCA)* v of V is the node where there are no LCA nodes on the path between v and v_i other than sign node v_i for any v_i. Let IL_i be the inverted list corresponding to keyword k_i. The ELCAs of K in T are given as

$$\begin{aligned} &elca(k_1, k_2, ..., k_n) = elca(IL_1, IL_2, ..., IL_n) \\ &= \{v | \exists s_1 \in IL_1, s_2 \in IL_2, ..., s_n \in IL_n \ s.t. \ (v = lca(s_1, s_2, ..., s_n) \\ &\wedge \forall i \in [1, n] \ \nexists x (x \in lca(IL_1, IL_2..., IL_n) \wedge v \prec x \wedge x \preceq s_i))\} \end{aligned}$$

Here, when given two nodes u and v, $v \prec u$ indicates that v is an ancestor of u and $v \preceq u$ indicates that v is an ancestor of u or $v = u$. Moreover, each s_i is called a *sign* node of k_i in node v. There can be multiple sign nodes of k_i in an ELCA node. Therefore, let Sgn_i be a set of sign nodes of k_i in v; we represent all sign nodes in v as $v.sign = (Sgn_1, Sgn_2..., Sgn_n)$.

The ELCAs of keywords "book john" in the XML tree in Figure 1 are $\{n(0.2)$, $n(0.3)$, $n(0.3.1.0)\}$. Here, $n(0.2).sign = (\{n(0.2)\}, \{n(0.2.1)\})$, $n(0.3).sign = (\{n(0.3)\}, \{n(0.3.1.1)\})$, $n(0.3.1.0).sign = (\{n(0.3.1.0)\}, \{n(0.3.1.0)\})$, and two sign nodes of node $n(0.3.1.0)$ do not appear as sign nodes of node $n(0.3)$. As seen above, there are no sign nodes that appear as sign nodes of different ELCAs.

(**Witness**). Let T be the XML tree and $K = \{k_1, k_2, ..., k_n\}$ be the input keyword set. Also, let r be an ELCA of K and s_i be a sign node of k_i of r. We define $witness(r, (s_1, s_2, ..., s_n))$, called the *witness* of K in T, as the pair of node r and the list of sign node s_i. The witness is visualy represented as a tree where node set $V = \{r, s_1, s_2, ..., s_n\}$ and edge set $E = \{(r, s_1), (r, s_2), ..., (r, s_n)\}$. Here, when node s_i has text node t_i, t_i is added to V and edge (s_i, t_i) is added to E. We call the tree given by this procedure a *witness tree*. We also denote the witness as $witness(dewey(r), (dewey(s_1), dewey(s_2), ..., dewey(s_n)))$ since each node is identified by an ID.

(**Witness pattern**). Let T be the XML tree and $K = \{k_1, k_2, ..., k_n\}$ be the input keyword set. Let $w = witness(r, (s_1, s_2, ..., s_n))$ be the witness of K in T; we define $(\lambda(r), (\lambda(s_1), \lambda(s_2), ..., \lambda(s_n)))$ to be the *witness pattern* of w and denote it as $\lambda(w)$. The witness pattern tree that visually represents $\lambda(w)$ as the tree is the witness tree removed from the Dewey order. Moreover, let w_1, w_2 be the witness of K in T; w_1 and w_2 are similar if and only if $\lambda(w_1) = \lambda(w_2)$.

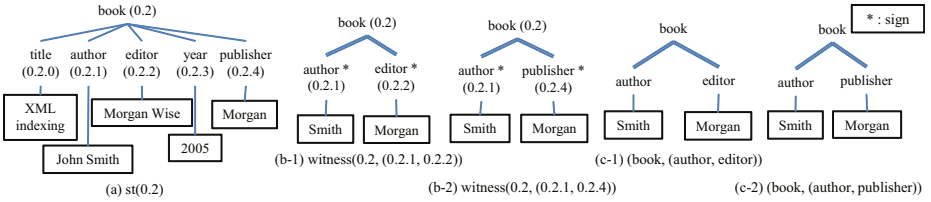


Fig. 2. Witness trees

Assume witness $w = witness(0.2, (0.2.1, 0.2.2))$, w' be $witness(0.2, (0.2.1, 0.2.4))$ of the keyword set "Smith Morgan" in Figure 1. Witness pattern $\lambda(w)$ is the $(book, (author, editor))$ represented as the tree in Figure 2(c-1). Witness pattern $\lambda(w')$ is $(book, (author, publisher))$ represented as the tree in Figure 2(c-2).

3.1 Classification of Search Results with Witness Patterns

We consider it would be better to classify the results into interpretations when we discover alternative queries. We omit some details on algorithms of classifying the results to make the paper short and we will alternatively introduce classification methods in this example. Consider the keyword set "Smith Morgan" applied to the data in Figure 1. The XML subtree $st(0.1.1)$ and its witness $w_1 = witness(0.1.1, (0.1.1.1, 0.1.1.2))$ are obtained at the beginning where the witness pattern of w_1 corresponds to the tree in Figure 3(a). Since a set of clusters is empty, a new cluster $C_1 = C((authors, (author, author)))$ is created and $st(0.1.1)$ is added to C_1. Next, the subtree $st(0.2)$ is obtained. Here, there are two witnesses $w_2 = witness(0.2, (0.2.1, 0.2.2))$ and $w_3 = witness(0.2, (0.2.1, 0.2.4))$. The witness pattern of w_2 and that of w_3 correspond to the trees in Figures 3(b) and (c) respectively. Since neither witness pattern is equal to C_1, new clusters $C_2 = C((book, (author, editor)))$ and $C_3 = C((book, (author, publisher)))$ are created, and then $st(0.2)$ is added to C_2 and C_3. Finally, the subtree $st(0.3)$ and its witness $w_4 = witness(0.3, (0.3.1.1, 0.3.2))$ are obtained where the witness pattern of w_4 corresponds to the tree in Figure 3(b). Since the witness pattern of w_4 is the same as of C_2, $st(0.3)$ is added to C_2. We have presented the classified results in Figure 3. Here, C_1, C_2, and C_3 are the clusters in Figures 3(a), (b), and (c) and the elements of each cluster are indicated as subtrees surrounded by a frame on the right-hand side.

4 Generation of Alternative Queries

Unlike IEQs in the relational model, alternative queries in XML keyword search have to consider interpretations of queries. We define a keyword set as an alternative query such as that where one of the clusters queried by the keyword set is equal to one of those queried by the input keyword set. The definition of an alternative query is as follows.

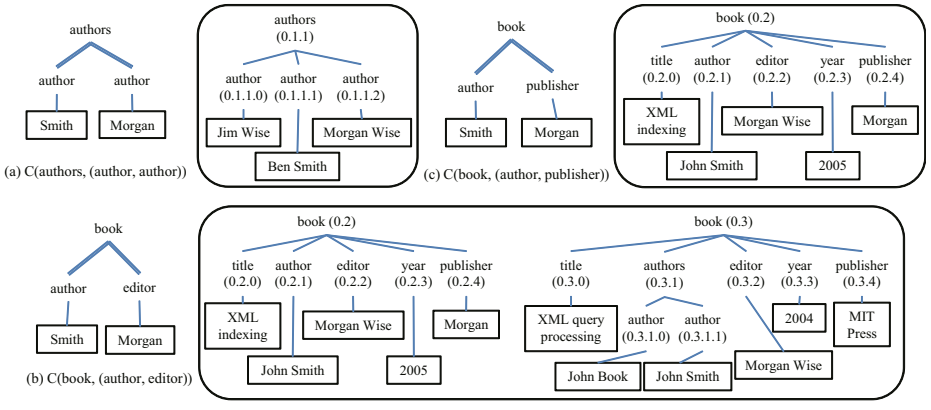


Fig. 3. Clusters, where each set of XML subtrees surrounded by a frame is element of cluster on left-hand side

(**Alternative Query**). Given input keyword set K and XML tree T, let $WP(T,K) = \{wp_1, wp_2, ..., wp_n\}$ be the witness pattern set of K in T, and the search results $R(K,T) = C(wp_1) \cup C(wp_2) \cup ... \cup C(wp_n)$ are obtained. When cluster $C_i = C(wp_i)$ is selected, we define keyword set Q that obtains $wp_j^{'} \in WP(T,Q)$ such that $C(wp_i) = C(wp_j^{'})$ as an alternative query of K for C_i. Alternative query Q_1 is smaller than alternative query Q_2, if and only if $Q_1 \subset Q_2$ holds.

Example 1. Consider keywords "Smith Morgan" applied to the XML tree in Figure 1. Then, search results $C_1 = \{st(0.1.1)\}$, $C_2 = \{st(0.2), st(0.3)\}$, $C_3 = \{st(0.2)\}$ are obtained. When cluster C_2 is selected, one of the alternative queries for C_i is $Q = \{wise, smith, xml\}$. In fact, two clusters $C_4 = \{st(0.1)\}$ and $C_5 = \{st(0.2), st(0.3)\}$ are obtained by applying keywords Q to the XML tree. Keyword set Q is the alternative query of keywords "Smith Morgan" for C_2 as C_2 and C_5 are equivalent.

Now, we describe the cluster which we want to generate alternative queries as a *positive subtree set* and describe each XML subtree in the positive subtree set as a *positive subtree*. We describe XML subtrees other than positive subtrees and rooted at the same element name with positive subtrees as *negative subtrees*, and describe a set of these as *negative subtree set*.

There may be no alternative queries in some situations. It is disadvantageous if users can not obtain any alternative queries from the viewpoint of supporting them understand search results. Thus, we suggest approximate alternative queries that obtain answers where the proportion of answers to positive subtrees is over given threshold value. An approximate alternative query is considered to have two meanings of approximation as follows: (1) how many positive subtrees does the approximate alternative query obtain in the positive subtree set, and (2) how many positive subtrees does the approximate alternative query obtain in the results queried by the approximate alternative query. The ratio indicating

(1) in the area of information retrieval can be considered to be precision and the ratio indicating (2) can be considered to be recall when we consider a positive subtree set as a set of answers. There are generally more queries when we generate approximate alternative queries than when we generate alternative queries, and the quality of the queries deteriorates. Thus, we have to pose users with ranked approximate alternative queries. First, we define precision and recall for approximate alternative queries.

(**Precision and Recall**). Given input keyword set K, and XML database T, let us consider $R(K,T) = \{C_1, C_2, ..., C_n\}$ to be the search results of K in T. When given approximate alternative query A for positive subtree set C_i, *precision* and *recall* of A are given as

$$precision = \frac{|C_i \cap C_j^{'}|}{|C_j^{'}|}, recall = \frac{|C_i \cap C_j^{'}|}{|C_i|}$$

Here, $C_j^{'} \in R(A,T)$ where the witness pattern of $C_j^{'}$ is equivalent to that of C_i.

Alternative queries can be considered as approximate alternative queries such that both the pricision and recall are 1.0. Thus, we discuss only the method of generating approximate alternative query in the rest of this paper.

4.1 Naive Generation of Approximate Alternative Queries

Let us consider generating approximate alternative queries from a set of XML subtrees in a cluster of classified XML search results.

Example 2. The results queried by keywords "Smith Morgan" in Figure 1 are classified into the three clusters in Figure 3. Let a positive subtree set be cluster $C_2 = C(book, \{author, editor\})$ in Figure 3(b); the positive subtree has the witness pattern represented as the tree in Figure 2(c-1). Here, a negative subtree set consists of two subtrees $st(0.0)$ and $st(0.1)$ where the XML subtrees are rooted at "book" nodes other than positive subtrees.

The simplest method of generating approximate alternative queries is to extract all terms that appear at least once in any positive subtrees, then filter any subset of the extracted terms that the precision and recall of the subset that exceed given threshold value. Here, it is a fact that recall r of approximate alternative query $\{w_1, w_2\}$ holds $r \leq min(p_{w_1}, p_{w_2})$ where p_{w_1} and p_{w_2} are occurrence frequencies in the positive subtree set when given two keywords w_1 and w_2. Therefore, the method of generating approximate alternative queries only extracts tagged keywords that exceed the threshold, then tallies any approximate alternative queries and calculate the precision and recall. The algorithm for generating approximate alternative queries is Algorithm 1.

4.2 Optimized Generation of Approximate Alternative Queries

The more the common keywords which appear in a positive subtree set in the naive method, the more exponential the numbers of alternative query candidates

Algorithm 1. Naive method of generating approximate alternative queries

Input: a positive subtree set PSS
Input: a witness pattern wp that all positive subtrees have
Input: threshold values θ_r and θ_p
1: $c = getAllCommonWords(PSS, \theta_r)$
2: get all Dewey inverted lists in c
3: **for all** subset s of c **do**
4: $E = getELCA(s)$
5: select cluster E_i from E such that the element name of the root node of the witness pattern of E_i corresponds to that of wp
6: calculate precision and recall of s
7: **end for**
8: return all keywords whose precision and recall exceed θ_p and θ_r respectively

that have to be verified. The algorithm generally verifies candidates by scanning all the inverted lists corresponding to each keyword in alternative query candidates including keyword w. Therefore, it is inefficient for the naive method to scan inverted lists corresponding to w whenever the algorithm verifies a candidate that contains w.

We propose an optimized method of scanning all inverted lists only once to hold all approximate alternative query candidates. Here, the method has to take structure information into account in processing. This is because there can be ELCAs obtained with the original query in ELCA semantics that are different from the ELCAs obtained by alternative query candidates. As a result, a set of keywords not satisfying the definition of an alternative query can be obtained.

The algorithm for generating approximate alternative queries is Algorithm 2. The algorithm executes two scans to generate alternative queries in parallel, called the *Cut off scan* (*C-scan*) and *Verification scan* (*V-scan*). The C-scan scans all positive subtrees to extract all tagged keywords that can be considered to be candidates of alternative queries. The V-scan tallies each tagged keyword set and the times each occurred in positive subtrees and negative subtrees for all possible alternative query candidates through the inverted lists and it reports tagged keyword sets that the precision and recall exceed given threshold.

The algorithm 2 extracts all tagged keywords such that occurrence frequencies exceed the threshold value from the positive subtree set. It stores the keyword set and its number of appearances in count table PCT when a keyword set appears in a positive subtree, and set these in count table NCT when it appears in a negative subtree. The algorithm calculates the precision, recall, and F-measure of each keyword set stored in PCT as a key when it finishes scanning the inverted lists, then adds the keyword set into list L if the precision and recall exceed the threshold values. Finally, it outputs all elements in L ordered by the F-measures in descending order.

Consider the query "smith morgan" applied to the XML tree in Figure 1. The method of classification returns three clusters in Figure 3: $\{C_1, C_2, C_3\}$ = {C(authors, {author, author}), C(book, {author, editor}), C(book, {author,

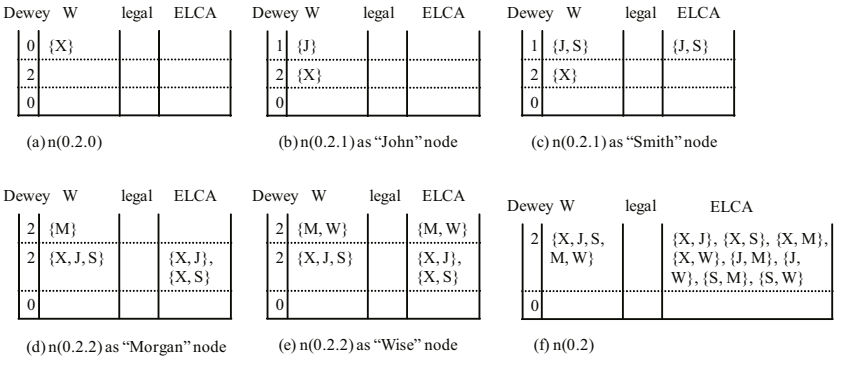


Fig. 4. States of stack, where X, J, S, M, W in stack entries stands for (xml,title), (john,author), (smith,author), (morgan,editor), (wise,editor) respectively

publisher})} = {{st(0.1.1)}, {st(0.2), st(0.3)}, {st(0.2)}}. When $PSS = C_2$ is selected and the element name of the root node of a positive subtree is "book", the algorithm obtains the inverted list corresponding to the "book" node and then $roots = [n(0.0), n(0.1), n(0.2), n(0.3)]$ are obtained (line 1). The algorithm scans the positive and negative subtrees rooted at a node in $roots$ to extract the set of tagged keywords that obtains the root as an LCA. As a result, it reports any tagged keyword set as alternative queries where each LCA in the keyword set appears in the root node of a positive subtree and does not appear in the root node of any negative subtrees. When the algorithm extracts the tagged keywords from all positive subtrees, $Common$ = {(xml, title), (john, author), (smith, author), (morgan, editor), (wise, editor)} are obtained (line 2). Here, we denote (xml, title) as X, (john, author) as J, (smith, author) as S, (morgan, editor) as M, and (wise, editor) as W. The top entry in the stack in Figure 4(a) shows X has appeared in the subtree rooted at $n(0.2.0)$. Smallest node v is $n(0.1.0)$ (line 4). When the smallest node is obtained, largest node r in the nodes that are smaller than v is $n(0.1)$ (line 6). If r is an ancestor of v, the algorithm obtains all nodes n that are descendants of r from inverted lists and pushes these into the stack (lines 8-10). All states of the stack when the algorithm processes $n(0.2.0)$, $n(0.2.1)$, and $n(0.2.2)$ in this order are shown in Figure 4(a), (b), and (d), respectively.

When a node is added to the stack, we utilize Algorithm 3. Consider that an empty stack and node $n(0.2.0)$ have been given. Three entries are initially pushed into the stack and Figure 4(a) indicates the initial state of the stack where W contains X in the top entry. Next, let us consider where the stack and "John" node $n(0.2.1)$ are given. Since the longest common prefix of 0.2.0 and 0.2.1 is 0.2, the top entry is popped out. When the top entry is popped out, information about W is passed to the top entry of the middle entry in the stack. After this, a new entry from the components of node $n(0.2.1)$ that are not among the longest common prefix is pushed into the stack. Figure 4(b) indicates the

Algorithm 2. Stack algorithm for generating approximate alternative queries

Input: a cluster $Cl = C(R, (e_1, e_2, ..., e_n))$, threshold values θ_p, θ_r

```
get roots as a keyword inverted list corresponding to R
Common = getCommonWordsWithParentName(Cl, θr)
stack = ∅, count table PCT = ∅, count table NCT = ∅, List L = ∅
v = getSmallestNode(Common)
while ( v != null ) {
  find the largest node r in the inverted list of root, which is smaller than v
  if ( r ⪯ v )
    for all ( r ⪯ n in all inverted lists )
      join(stack, n)
    for all ( e in stack.getELCACandidates(r) )
      if ( st(r) is in Cl )
        num = PCT.get(e)
        PCT.set(e, num + 1)
      else
        num = NCT.get(e)
        NCT.set(e, num + 1)
}
for all ( k in PCT.key )
  calculate precision, recall, F-measure
  if ( precision ≥ θp && recall ≥ θr )
    L.add(k)
output all elements in L order by F-measure
```

state of the stack after the algorithm has processed node $n(0.2.1)$. Each figure in Figure 4 other than Figure 4(f) indicates the state of the stack after the node in the caption has been processed. Figure 4(f) indicates the state of the stack when the top entry is popped out from the state of the stack in Figure 4(e). When we obtain all sets of keywords whose ELCA is $n(0.2)$ from the stack, all entries are popped out until $n(0.2)$ is popped out. When the top entry of the stack whose state is in Figure 4(f) is popped out, $ELCA$ in the popped entry contains $\{X, J\}$, $\{X, M\}$ and so on. Thus, $\{X, J\}$, $\{X, M\}$, and any combination of $ELCA$ in the popped entry are sets of keywords to obtain ELCA.

5 Experiments

This section explains our evaluation of the performance of the proposed and naive methods. We carried out the evaluation by comparing the calculation times using the naive and optimized methods to generate approximate alternative queries.

We implemented ELCA, the classification method we propose, and both naive and optimized methods to generate approximate alternative queries in Java. The experiments were done on a 2.67 GHz Intel Core i7 machine with 6 GB RAM running 64 bit Windows 7. We tested the 670 MB DBLP [2] as a real dataset and

[2] http://www.informatik.uni-trier.de/~ ley/db/

Algorithm 3. $join(stack, n)$ method for ELCA

```
1: p = lca(stack, n)
2: while ( stack.size > p ) {
3:   entry = stack.pop()
4:   if ( !stack.isEmpty() )
5:     parentEntry = stack.pop()
6:     D1 = entry.W \ parentEntry.W
7:     D2 = parentEntry.W \ entry.W
8:     for all ( d in D1 )
9:       for all ( w in entry.W \ D1 )
10:        parentEntry.legal.add(w(d))
11:    for all ( d in D2 )
12:      for all ( w in entry.W \ D2 )
13:        parentEntry.legal.add(w(d))
14:    for all ( w1(d1) in entry.legal )
15:      for all ( w2(d2) in parentEntry.ligal )
16:        if ( w1 == d2 && w2 == d1 )
17:          remove w1(d1) from entry.legal
18:          remove w2(d2) from parentEntry.legal
19:    parentEntry.legal.add(entry.legal)
20:    add direct product of entry.W and parentEntry.W to parentEntry.ELCA
21:    parentEntry.W.add(entry.W)
22:    stack.push(parentEntry)
23: }
24: for ( p < j ≤ n.length )
25:   stack.push(n[j][][])
26: stack.top.W.add(n)
```

selected the 20 queries in Table 1 where Q_1 to Q_4 are pairs of two author names, Q_5 to Q_8 are pairs of an author and a specific keyword, Q_9 to Q_{12} are pairs of an author and an entity name, Q_{13} to Q_{16} are keywords that can be considered to be queried in Web search, and Q_{17} to Q_{20} are keywords that can be considered to be queried in Web search adding an entity name. We set threshold values of $\theta_p = 0.6$ and $\theta_r = 0.6$. # in Table 1 represents the number of common words that appear in positive subtree sets over θ_r. We regard the largest cluster in other clusters as a positive subtree set for each query. We utilized MySQL to store the extended Dewey inverted lists in the dataset.

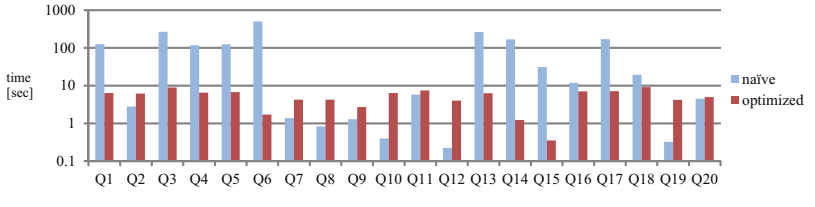


Fig. 5. Response time of approximate alternative query generation

Table 1. Query sets for experiments

	input keywords	#
Q_1	Shimizu, Yoshikawa	7
Q_2	Sakurai, Yoshikawa	6
Q_3	Zhifeng, Bao, Jiaheng Lu	11
Q_4	Toba, Milo, Dan, Suciu	12
Q_5	Toshiyuki, Shimizu, XML	7
Q_6	Katsumi, Tanaka, credibility	12
Q_7	Taro, Saito, XML	5
Q_8	Mizuho, Iwaihara, XML	5
Q_9	Jim, Gray, article	2
Q_{10}	Qiang, Ma, inproceedings	2

	input keywords	#
Q_{11}	Tok, Wang, Ling, inproceedings	3
Q_{12}	Toshiyuki, Shimizu, inproceedings	2
Q_{13}	VLDB, 2009, XML	12
Q_{14}	XML, data, mining	3
Q_{15}	query, optimization, database	3
Q_{16}	query, expansion, SIGIR	3
Q_{17}	XML, data, mining, inproceedings	3
Q_{18}	query, optimization, inproceedings	3
Q_{19}	XML, classification, inproceedings	2
Q_{20}	query, expansion, inproceedings	2

5.1 Performance

First, we evaluate all queries in Table 1 and present the summarized results in Figure 5. Figure 6 compares the performance of the naive and optimized methods in reading inverted lists. Figure 7 compares the performance of the naive and optimized methods with V-scan. From Figure 5, we can see the optimized method is faster than the naive one for Q_1CQ_3, Q_4, Q_5CQ_6CQ_{13}, Q_{14}CQ_{15}CQ_{16}CQ_{17}, and Q_{18} but the naive method is faster than the optimized one for the others. Now, let Q_{fast} be the set of queries where the optimized method is faster than the naive one and Q_{slow} be the rest. The naive method takes a great deal time to compute with V-scan for each query in Q_{fast} from Figure 7. Otherwise, it does not take a long time for each query in Q_{slow}. The optimized method, on the other hand, takes a long time to read inverted lists for Q_{slow} queries from Figure 6. Most queries are fast with the naive method from Figure 6 in the details on the reading time of inverted lists. This is because the optimized method reads the inverted list of the root node of the witness pattern in addition to inverted lists the naive method reads. We have compared the performance with V-scan in Figure 7, where the optimized method is faster than the naive one for all queries. This is because the optimized method scans each inverted list only once; however, the naive method repeatedly scans each inverted list.

Finally, we present a point diagram where the X-axis represents the number of common words appearing in a positive subtree set and the Y-axis represents the performance in Figure 8. It seems that the more words there are, the longer it takes the naive methods to calculate the generation of approximate alternative queries. However, performance of the optimized method is nearly constant. This is because it scans all the inverted lists only once; however, the calculations times with the naive method exponentially increase as the number of common words increases since the naive method computes the number of all possible combinations of tagged keywords to calculate precision and recall. The optimized method is often slow when the number of common words appearing in a positive subtree set is small. In these cases, the naive method is fast because of the number of approximate alternative queries that it has to compute. However, let

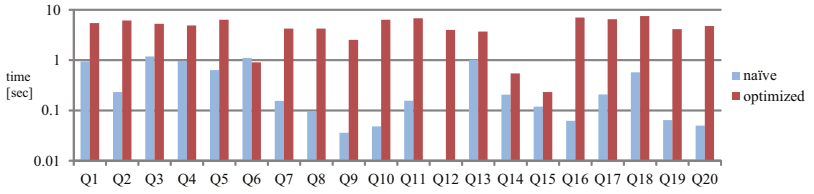


Fig. 6. Response time of naive method and optimized method in reading inverted lists

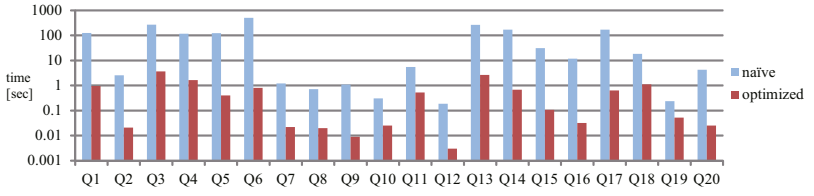


Fig. 7. Response time of naive method and optimized method in V-scan

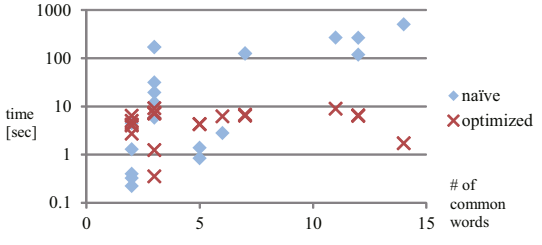


Fig. 8. Relationship of the number of common words and response time

us consider queries such as Q_9, Q_{10}, Q_{12}, Q_{19}, and Q_{20} where there are two common words, in which the naive method performs better. The common words for all queries mentioned above are the subset of input queries, which means the approximate alternative queries generated by these keywords is not useful because they give us only known information. More common words are needed to generate useful approximate alternative queries, and the optimized method outperforms the naive methods for such cases.

6 Conclusion

This paper addressed the problem with the diversity of interpretations in XML keyword search to classify results using witnesses and witness patterns. Witnesses and witness patterns are considered to correspond to the interpretation of XML keyword search since these concepts can capture the features of XML subtrees. We defined the notion of alternative queries and approximate alternative queries. We proposed a method of generating approximate alternative

queries and an optimized algorithm to generate alternative queries in parallel. Finally, we demonstrated the superiority of the optimized method in our experiments. The optimized method is faster than the naive method especially for the cases that there are many words which could be utilized as a component of approximate alternative queries.

References

1. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked keyword search over XML documents. In: SIGMOD Conference, pp. 16–27 (2003)
2. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest LCAs in XML databases. In: SIGMOD Conference, pp. 527–538 (2005)
3. Li, Y., Yu, C., Jagadish, H.V.: Schema-Free XQuery. In: VLDB, pp. 72–83 (2004)
4. Huang, Y., Liu, Z., Chen, Y.: Query biased snippet generation in XML search. In: SIGMOD Conference, pp. 315–326 (2008)
5. Liu, Z., Sun, P., Chen, Y.: Structured search result differentiation. PVLDB 2(1), 313–324 (2009)
6. Andritsos, P., Miller, R.J., Tsaparas, P.: Information-theoretic tools for mining database structure from large data sets. In: SIGMOD Conference, pp. 731–742 (2004)
7. Wu, W., Reinwald, B., Sismanis, Y., Manjrekar, R.: Discovering topical structures of databases. In: SIGMOD Conference, pp. 1019–1030 (2008)
8. Motro, A.: Intensional answers to database queries. IEEE Trans. Knowl. Data Eng. 6(3), 444–454 (1994)
9. Tao, Y., Yu, J.X.: Finding frequent co-occurring terms in relational keyword search. In: EDBT, pp. 839–850 (2009)
10. Tran, Q.T., Chan, C.Y., Parthasarathy, S.: Query by output. In: SIGMOD Conference, pp. 535–548 (2009)
11. Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective XML keyword search with relevance oriented ranking. In: ICDE, pp. 517–528 (2009)
12. Schmidt, A., Kersten, M.L., Windhouwer, M.: Querying XML documents made easy: Nearest concept queries. In: ICDE, pp. 321–329 (2001)
13. Liu, Z., Chen, Y.: Identifying meaningful return information for XML keyword search. In: SIGMOD Conference, pp. 329–340 (2007)
14. Supasitthimethee, U., Shimizu, T., Yoshikawa, M., Porkaew, K.: XSemantic: An extension of LCA based XML semantic search. IEICE Trans. 92-D(5), 1079–1092 (2009)
15. Robertson, S.E.: On term selection for query expansion. J. Doc. 46, 359–364 (1991)
16. Qiu, Y., Frei, H.P.: Concept based query expansion. In: SIGIR, pp. 160–169 (1993)
17. Amer-Yahia, S., Lakshmanan, L.V.S., Pandit, S.: FleXPath: Flexible structure and full-text querying for XML. In: SIGMOD Conference, pp. 83–94 (2004)
18. Liu, C., Li, J., Yu, J.X., Zhou, R.: Adaptive relaxation for querying heterogeneous XML data sources. Inf. Syst. 35(6), 688–707 (2010)
19. King, J.J.: QUIST: A system for semantic query optimization in relational databases. In: VLDB, pp. 510–517 (1981)
20. Freytag, J.C.: A rule-based view of query optimization. In: SIGMOD Conference, pp. 173–180 (1987)

K-Graphs: Selecting Top-k Data Sources for XML Keyword Queries

Khanh Nguyen and Jinli Cao

Department of Computer Science and Computer Engineering
La Trobe University, Melbourne, Australia
{tuan.nguyen,j.cao}@latrobe.edu.au

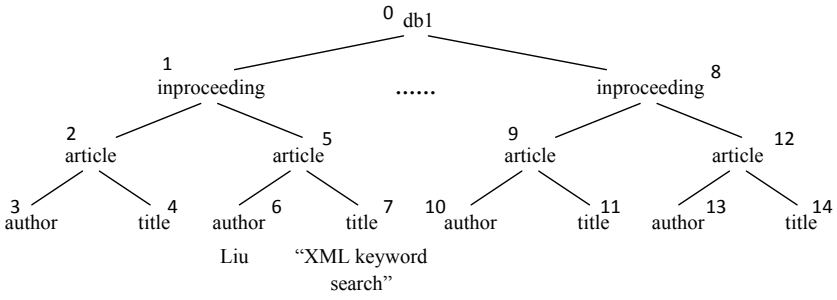
Abstract. Most of existing approaches on XML keyword search focus on querying over a single data source. However, searching over hundreds or even thousands of (distributed) data sources by sequentially querying every single data source is extremely costly, thus it can be impractical. In this paper, we propose an approach for selecting top-k data sources to a given query in order to avoid the high cost of searching numerous, potentially irrelevant data sources. The proposed approach can efficiently select top-k mostly relevant data sources without querying over the data sources. We propose a ranking function for measuring the strength of correlation between keywords in a data source and summarize the data sources as keywords correlation graphs (K-Graphs). The top-k relevant data sources will be selected by estimating the relevance of corresponding K-Graphs to the query. Experimental results show that the approach achieves good performance with a variety of experimental parameters.

1 Introduction

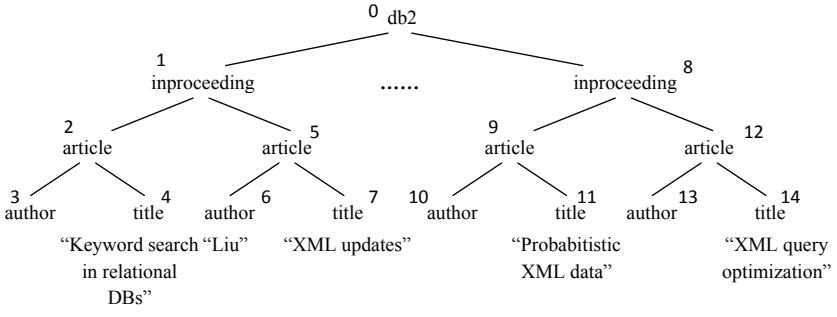
The Extensible Markup Language (XML) has become a *de facto* standard for representing and exchanging data, resulting in the proliferation of XML documents distributed over the internet. Traditionally, XML data are retrieved using structured query languages such as XPath and XQuery, in which users have to learn both data schema and query languages in order to effectively issue queries. Since the data schema and the query languages may be complex, retrieving XML data using XPath/XQuery languages is usually limited to advanced users. In that context, keyword-based search over XML data has been proposed as a mean to liberate users from the learning curve of the structured query languages, thus attracted significant attention of researchers from both fields of information retrieval and databases.

Querying XML data using keyword-based search has been widely studied in literature [1–7], however most of existing approaches focus on query processing over a single data source. Searching through hundreds or even thousands of data sources by sequentially querying each data source is extremely costly and may not be practical, while efficient query processing even in single data source is a challenging problem [8–12]. Efficient query processing over a system which integrates numerous data sources is definitely much more challenging. Thus, how

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 425–439, 2011.



(a) T_1



(b) T_2

Fig. 1. An example of bibliographic data sources

to address the problem of query processing over multiple XML data sources is a challenging issue in practice.

In the context of information retrieval (IR), selecting most useful data sources from a large number of sources has been studied [23–25]. The common approach is to summarize each data source as term statistics (e.g., term frequency and inverse document frequency). Given a query, the system can select the most appropriate data sources by measuring the relevant degree between the summarized statistics and the query. In this case, the statistics act as data sources' summaries for fast filtering non-promising sources which in turn accelerates the overall query processing.

However, applying IR techniques to the context of XML data may be inadequate for the following reasons. First, although using term statistics is effective in IR field, the term statistics solely are not effective to measure the relevance of an XML source to a given query. This is because the structures of XML data (or schema) convey rich semantics, hence they should be considered when measuring the relevance of a data source, besides the term statistics. Second, the term occurrences solely in an XML data source do not guarantee the appearance of relevant results in that data source. In other words, the relevance of an XML data source also depends on how closer the relationship of query keywords in that data source. For demonstration, let us consider two data sources T_1 and T_2

in Figure 1 and assume that a user desires to search for articles of "*Liu*" about "*XML keyword*" by issuing query $Q = \{Liu, XML, keyword\}$. We can observe that T_1 has one "relevant" result to query Q. On contrast, T_2 does not contain any "relevant" result to Q.

Table 1. *KF-Summary* for T_1 and T_2

keyword	**frequency**	
	T_1	T_2
Liu	1	1
XML	1	3
keyword	1	1
...	...	...

However, based on the keyword frequency summaries (denoted as *KF-summary* in this paper) shown in Table 1, T_2 will be selected over T_1. This is because the frequency of the query keywords in T_2 is higher than the frequency of these keywords in T_1. For further illustration, we will deeply study the low precision of *KF-summary* for data sources selection in the experimental section. For the above reasons, we can conclude that the relevance of an XML data source is not only decided by keyword frequency, but more importantly it depends on how closer the relationship between the query keywords in each data source.

In this paper, we propose an approach for processing a keyword query over multiple data sources. The proposed approach selects top k relevant data sources in which the query will be forwarded, where k is an users' selected parameter. The contributions in this paper are summarized as follows:

- We propose an approach to select top-k XML data sources for keyword queries without querying the data sources. To obtain this aim, we first propose a method for evaluating the relationships between keywords and summarize the data sources as K-Graphs which maintain those keyword relationships.
- We define criteria for ranking the relevance of the data sources to a given keyword query by estimating the correlation of query keywords in the corresponding K-Graphs of the data sources in order to fast select top-k ranked data sources.
- We conducted experiments using real-life data set for evaluating the performance of the proposed approach. The experimental analysis shows that our approach has good performance in a variety of experimental parameters.

The rest of this paper is organized as follows. Section 2 is an overview of our approach. Section 3 presents our techniques for summarizing data sources. In section 4, we propose our ranking functions for estimating the relevance of a data source to a given query and top-k data source selection. We study the experimental results in section 5. Section 6 discusses related work. Finally, conclusions are given in section 7.

2 Overview of Approach

2.1 Data Models and Queries

We consider a data source as an XML document and we model each data source as a tree. An XML data tree is defined as $T = (V_T, E_T)$ where V_T is a finite set of nodes, representing elements and attributes of the data tree T; E_T is set of directed edges where each edge $e(v_1, v_2)$ represents the parent-child relationship between the two nodes $v_1, v_2 \in V_T$. We assume that all values appear in the leaf nodes. Figure 1, for instance, represents two XML data trees T_1 and T_2 containing bibliographic information.

A keyword query is a set of different terms, denoted by $Q = \{k_1, k_2, \ldots, k_q\}$. We consider the AND-semantics for the query. A query result must contain at least one occurrence of each term $k_i \in Q$.

2.2 Overview of Approach

We consider a set of XML data sources $\mathcal{T} = \{T_1, \ldots, T_N\}$. Given a keyword query $Q = \{k_1, \ldots, k_q\}$, we would like to rank the data sources in $\mathcal{T}$ based on their usefulness to the query Q. Basically, the usefulness of a data source can be computed as the total score of all its results to query Q. However, this approach can overestimate data sources containing numerous results over other data sources consisting of high quality results. To be balanced, the relevance of a data source T_i to query Q is frequently evaluated as the total score of its top k results, where k is a selected parameter from users.

$$\mathsf{score}(T_i, Q) = \sum_{i=1}^{k} \mathsf{score}(R_i, Q), \tag{1}$$

where R_i is the i-th top result of Q in T_i and $\mathsf{score}(R_i, Q)$ is the relevant score of R_i to Q.

Ideally, the data sources should be ranked in descending order of their scores calculated according to Equation 1. However, to calculate the ideal scores of the data sources, the system needs to execute the query over all the data sources. Because the number of data sources being searched can be very high and the data sources can be very large, searching through all those data sources can be very time consuming, thus it may be impractical.

The aim of our work is to propose an approach which can efficiently and effectively select top-k data sources amongst potentially numerous data sources without querying over the data sources. To obtain this aim, we construct summaries for the data sources *off-line* and select the useful data sources by calculating the relevant scores of the summaries to the query *online*. Each summary (namely K-Graph in this paper) of a data source stores relationships between keywords appearing in the data source, where the keyword relationships are evaluated by our ranking function, considering both content relevance and structure relevant factors. Finally, we present two methods for estimating the relevance of data sources to a given query based on the constructed summaries.

3 Keyword Correlation Graph (K-Graph)

We summarize a data source as a keyword correlation graph (or K-Graph for short). The K-Graph measures correlations between keywords in the data source. Nodes of the graph are labeled by keywords appearing in the data source. The edge between two nodes k_i and k_j is marked by distinct integer numbers which indicate lengths of the paths connecting k_i and k_j in the data source. As two keywords can be connected through paths with different distances, we will present, in the following subsection, our method for evaluating the correlation between keyword k_i and k_j at specific distance d as well as the correlation between k_i and k_j in the K-Graph. Given two nodes n_i and n_j which contain two keywords k_i and k_j, we define the score of k_i and k_j with respect to n_i and n_j as

$$\mathsf{score}(k_i, k_j, n_i, n_j) = \frac{\mathsf{weight}(n_i, k_i) + \mathsf{weight}(n_j, k_j)}{\mathsf{dist}(n_i, n_j) + 1} \tag{2}$$

where $\mathsf{dist}(n_i, n_j)$ is the distance of the path connecting two nodes n_i and n_j which measures how strong the relationship between the two nodes, in the sense that the closer distance between the two nodes indicates their stronger relationship. $\mathsf{weight}(n_i, k_i)$ measures the content relevance of node n_i with respect to keyword k_i.

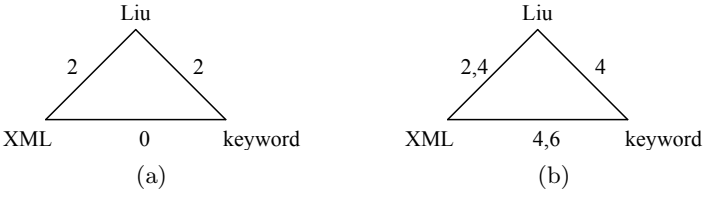


Fig. 2. K-Graphs of keywords $\{Liu, XML, keyword\}$ in T_1 (a) and T_2 (b)

In order to calculate $\mathsf{weight}(n_i, k_i)$, we employ the standard $tf * idf$ from the information retrieval field. The $tf * idf$ measures the content relevance of a document to a keyword query using both *term frequency* (i.e., how many times a term appears in a document) and *inverse document frequency* (i.e. inverse of how many documents contain the term). In order to apply it to XML data scenarios, we make some following adaptations: firstly, *term frequency* (tf) of a term k_i in node n_i is the number of occurrences of k_i in node n_i and as [13] we assume that term frequency (tf) is always equal to 1; secondly, we adaptively define *inverse element frequency (ief)* of a term t as the total number N of element in the XML data tree over the number N_t of elements that contain the term t, i.e. $ief_t = \frac{N}{N_t}$.

Based on the above adaptations, the weight of keyword k_i in node n_i is calculated as

$$\mathsf{weight}(n_i, k_i) = \log_2(1 + tf_{k_i}) \log_2 ief_{k_i} = \log_2 ief_{k_i} \tag{3}$$

Considering data tree T_1, for instance, we have:

$$\mathsf{weight}(7, \text{“}XML\text{”}) = \mathsf{weight}(7, \text{“}keyword\text{”}) = \log_2(\frac{15}{1}) = 3.91$$

Similarly, from data tree T_2 we have:

$$\mathsf{weight}(7, \text{“}XML\text{”}) = \mathsf{weight}(11, \text{“}XML\text{”}) = \mathsf{weight}(14, \text{“}XML\text{”}) = \log_2(\frac{15}{3}) = 2.32$$

$$\mathsf{weight}(4, \text{“}keyword\text{”}) = \log_2(\frac{15}{1}) = 3.91$$

Thus, from the data tree T_1 we have:

$$score(\text{“}XML\text{”}, \text{“}keyword\text{”}, 7, 7) = \frac{\mathsf{weight}(7, \text{“}XML\text{”}) + \mathsf{weight}(7, \text{“}keyword\text{”})}{0+1} = 3.91 + 3.91 = 7.82$$

Similarly, from the data tree T_2 we have:

$$\mathsf{score}(\text{“}XML\text{”}, \text{“}keyword\text{”}, 7, 4) = \frac{\mathsf{weight}(7, \text{“}XML\text{”}) + \mathsf{weight}(4, \text{“}XML\text{”})}{4+1} = \frac{3.91 + 2.32}{5} = 1.25$$

$$\mathsf{score}(\text{“}XML\text{”}, \text{“}keyword\text{”}, 11, 4) = \frac{\mathsf{weight}(11, \text{“}XML\text{”}) + \mathsf{weight}(4, \text{“}XML\text{”})}{6+1} = \frac{2.32 + 3.91}{7} = 0.89$$

$$\mathsf{score}(\text{“}XML\text{”}, \text{“}keyword\text{”}, 14, 4) = \frac{\mathsf{weight}(14, \text{“}XML\text{”}) + \mathsf{weight}(4, \text{“}keyword\text{”})}{6+1} = \frac{3.91 + 2.32}{7} = 0.89$$

3.1 Keyword Correlation at Specific Distance *d*

Let two nodes S_i and S_j are the sets of nodes containing keywords k_i and k_j respectively. We define the correlation between the two keywords k_i and k_j at specific distance d in a data tree T as

$$\mathsf{corr}(k_i \leftrightsquigarrow^d k_j) = \sum_{n_i \in S_i, n_j \in S_j : \mathsf{dist}(n_i, n_j) = d} \frac{\mathsf{score}(k_i, k_j, n_i, n_j)}{f_d(k_i, k_j)} \quad (4)$$

where $\mathsf{score}(k_i, k_j, n_i, n_j)$ is calculated as Equation 2 and $f_d(k_i, k_j)$ is the number of d-distance paths connecting the two keyword k_i and k_j.

In this way, we measure the correlation between the two keywords k_i and k_j at distance d as the average score of all paths connecting k_i and k_j at distance d. For illustration, the correlation between two keywords "XML" and "keyword" in the data tree T_1 at distance 0 can be computed as:

$$\mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow^0 \text{“}keyword\text{”}) = \mathsf{score}(\text{“}XML\text{”}, \text{“}keyword\text{”}, 7, 7) = 7.82$$

Similarly, from the data tree T_2 we have

$$\mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow^4 \text{“}keyword\text{”}) = \mathsf{score}(\text{“}XML\text{”}, \text{“}keyword\text{”}, 7, 4) = 1.25$$

$$\begin{aligned}&\mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow^6 \text{“}keyword\text{”})\\ &= \frac{\mathsf{score}(\text{“}XML\text{”}, \text{“}keyword\text{”}, 11, 4) + \mathsf{score}(\text{“}XML\text{”}, \text{“}keyword\text{”}, 14, 4)}{2}\\ &= \frac{0.89 + 0.89}{2} = 0.89\end{aligned}$$

Given any two keywords k_i and k_j in the XML data tree T, we can see that the maximum distance, d_{max} between k_i and k_j can be as twice as the hight of the data tree T, i.e., $d_{max} \leq 2 * h(T)$ where $h(T)$ is the hight of the data tree T. For instance, both data trees T_1 and T_2 have the height of 4, thus the maximum distance between any two keywords in those data trees is always less than or equal to 8.

3.2 Keyword Correlation

From the K-Graphs in Figures 2 we can see that two keywords in a data source can be connected at various distances with different correlation strengths, as measured by Formula 4. Now we define the total correlation between two keywords k_i and k_j in a K-Graph as follows.

Definition 1 (Keyword Correlation). *Let ω is the maximum length of the path between any two keywords k_i and k_j and k be the maximum number of results expected from an XML tree T. For each distance d, $f_d(k_i, k_j)$ is the frequency of d-distance paths connecting the two keywords. The keyword correlation between k_i and k_j represents strength of the relationship between the two keywords in the data tree T with respect to k. If $\sum_{d=0}^{\omega} f_d(k_i, k_j) \leq k$,*

$$\mathit{corr}(k_i \leftrightsquigarrow k_j) = \sum_{d=0}^{\omega} \mathit{score}(k_i \leftrightsquigarrow^d k_j) * f_d(k_i, k_j) \tag{5}$$

Otherwise, if $\sum_{d=0}^{\omega} f_d(k_i, k_j) \geq k$, we have $\exists \omega' \leq \omega, \sum_{d=0}^{\omega'} f_d(k_i, k_j) \geq k$ and $\sum_{d=0}^{\omega'-1} f_d(k_i, k_j) \leq k$,

$$\begin{aligned}\mathit{corr}(k_i \leftrightsquigarrow k_j) = &\sum_{d=0}^{\omega'-1} \mathit{score}(k_i \leftrightsquigarrow^d k_j) * f_d(k_i, k_j)\\ &+ \mathit{score}(k_i \leftrightsquigarrow^{\omega'} k_j) * (k - \sum_{d=0}^{\omega'-1} f_d(k_i, k_j))\end{aligned} \tag{6}$$

In other words, the keyword correlation measures the total scores of up to top-k correlations for each pair of keywords in an XML data source. A data source with a higher relationship score for a given pair of keywords will generate better results. The reason we set the upper-bound of the number of results k, is to enable a user to control the quality of one data source.

Let $k = 2$ be the maximum number of expected results, we can calculate the correlation between keyword pair ("XML", "$keywords$"), for instance, in the data tree T_1 as

$$\mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow \text{“}keyword\text{”}) = \mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow^{0} \text{“}keyword\text{”}) = 7.81$$

Similarly, the correlation of that pair of keywords in the data tree T_2 is

$$\begin{aligned}\mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow \text{“}keyword\text{”}) &= \mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow^{4} \text{“}keyword\text{”}) + \\ &\quad \mathsf{corr}(\text{“}XML\text{”} \leftrightsquigarrow^{6} \text{“}keyword\text{”}) \\ &= 1.25 + 0.89 = 2.14\end{aligned}$$

From that we can see that the correlation between keywords "XML" and "keyword" in the data tree T_1 is much stronger than that correlation in the data tree T_2, although the frequencies of those keywords in the data tree T_2 are higher than those frequencies in the data tree T_1.

3.3 Reducing Size of K-Graphs

We aware that indexing all pairs of keywords at all possible distances can result in extremely large K-Graphs. In addition, some work [1] in literature has pointed out that not all pair of keywords in an XML tree are meaningful related, especially those keywords which appear far way from each other. Thus, to reduce the size of the K-Graphs, we allow users (i.e., system administrators) limit the maximum allowed distance of between keywords in the K-Graphs, or they can define the meaningful relationship between keywords to be indexed, i.e., two keywords are meaningful related if the path connecting them does not contain two nodes with the same label [1]. We plan the study of this issue as future work.

4 Top-k Data Source Selection

In this section, we present our strategy for measuring and selecting appropriate data sources for a given query.

4.1 Estimating Relevant Scores of Data Sources

We estimate the relevance of a data source to a given keyword query, considering AND semantics which was popularly used in existing work [1–7]. This semantics requires each result must contain all query keywords.

Given a keywords query $Q = \{k_1, k_2, \ldots, k_q\}$, and a set of XML data sources $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$, we estimate the relevance of each data source T in $\mathcal{T}$ based on one of the following equations:

$$CORR\text{-}S(T, Q) = \sum_{\{k_i, k_j\} \subseteq Q, i<j} corr(k_i \leftrightsquigarrow k_j) \tag{7}$$

Equation 7 measures the relevance of a data source to a keyword query as sum of the strength of the relationships of every query keyword pair in T.

$$CORR\text{-}P(T, Q) = \prod_{\{k_i, k_j\} \subseteq Q, i<j} corr(k_i \leftrightsquigarrow k_j) \tag{8}$$

Equation 8 measures the relevance of a data source to a keyword query as product of the strength of the relationships of every query keyword pair in T.

4.2 Selecting Top-k Data Sources

Based on data source score calculated by Formula 5 and 6, we can effectively rank a set of XML trees $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ for a given keyword query Q. Specifically, the ranking is a mapping from $\mathcal{T}$ to $\{1, 2, \ldots, n\}$, such that $\mathsf{rank}(T_i, Q) < \mathsf{rank}(T_j, Q)$, iff $\mathsf{score}(T_i; Q) \geq \mathsf{score}(T_j; Q)$, where $\mathsf{score}(T_i; Q)$ denotes the degree of relevance of T_i to a given keyword query Q and are estimated using Formula 7 or 8. With a user provided number k, we can select the top k data sources with highest ranks.

5 Experimental Evaluation

In order to evaluate the performance of our proposed approach, we use real life DBLP [14] data set to generate 87 XML data sources by decomposing the whole data set according to different bibliographic types such as *inproceedings*, *articles*, *books*. Since some of the decomposed data sources (e.g., *inproceedings*, *articles*) are rather large, we decompose them further to derive a total of 87 data sources. There is no overlap between different generated data sources. The numbers of elements of the data sources are approximately similar. We implemented the algorithms using Java programming. We used Oracle Berkeley DB [15] as a tool for creating indexes.

5.1 Evaluation Metrics

To evaluate the effectiveness, we compare our approach with the *brute force selection*, which sends the given query to all the data sources for calculating top-k results, where the results of tested queries are identified by the query semantics proposed in [9]. The "real" score of each data source is calculated as Formula 1. We score a result by using a ranking function that is widely adopted in the XML keyword search scenario, as work [16]. We assign the local score of a keyword

k_i in node n_i as Equation 3 and select SUM as the aggregation function. Note that the execution time of such a *brute force selection* is orders of magnitude longer than that of our approach. To evaluate the performance of our approach, we employ two IR metrics *recall*, *precision* that were used for evaluating text collection selection algorithms by [17]. The *Recall(R)* is defined as,

$$R(K) = \frac{\sum_{T \in Top_K(S)} score(T, Q)}{\sum_{T \in Top_K(R)} score(T, Q)},$$

where $Top_K(S)$ and $Top_K(R)$ represent the K data sources with highest ranks using summary-based rankings and real rankings of data sources respectively.

Note that $score(T, Q)$ is the real score generated according to Formula 1. The recall definition compares the accumulated score of the top K data sources selected based on the summaries of the data sources against the total available score when we select top K data sources according to the real ranking. *Precision(P)* is defined as,

$$P(K) = \frac{|\{T \in Top_K(S) | score(T, Q) > 0\}|}{|Top_K(R)|},$$

This gives the fraction of the top K data sources in the estimated ranking that have non-zero score.

We compare the effectiveness of our approach against the keyword frequency summary, which is typically used as the summary of textual document collection for text collections selection [18], denoted as *KF-summary*. The *KF-summary* of each XML data source is a list of keywords which appear in the data source associated with their frequencies, i.e., the number of elements that contain the keyword. Based on the *KF-summary*, we estimate the score of a data source T for a given query $Q = \{k_1, \ldots, k_q\}$ in two ways. One is by summing the frequencies of all query keywords in T, i.e.,

$$KF\text{-}SUM(T, Q) = \sum_{i=1}^{q} freq(k_i) \tag{9}$$

The other is to take the product of the frequencies, i.e.,

$$KF\text{-}PROD(T, Q) = \prod_{i=1}^{q} freq(k_i) \tag{10}$$

The experiments involve two parameters: the *number of query keywords* and *the number K of selected data sources*.

5.2 Results and Analysis

To compare our approach with *KF-summary* approach, we tested 50 queries consisting of two to five keywords. The keyword queries are composed of randomly selected keywords from the data sources. The score of each data source

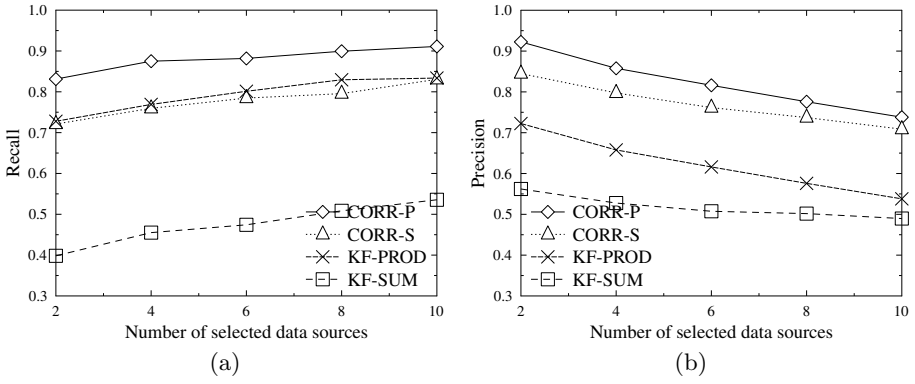


Fig. 3. Recall and precision of queries versus number of selected data sources

is calculated based on the sum of its top-10 highest score results. The recall and precision of each query types are calculated as the average of the experimental results of 50 tested queries.

Figure 3a shows that our method CORR-S outperforms KF-summary method KF-SUM, while CORR-P outperforms KF-PROD in the *recall* metric. Regarding our approach, the KF-PROD performs slightly better than CORR-S. The observed reason is that CORR-S sometimes ranked data sources partially containing query keywords higher. This is also the reason of low performance of KF-SUM in comparison with KF-PROD. In term of precision, Figure 3b indicates that our both methods achieve higher performance than KF-summary methods. Especially, our method CORR-P performs better than all other ones at approximately from 15% to 20%.

Effects of the number of query keywords. We also studied the effects of the number of query keywords to the recall and precision of our approach. Overall, figures 4a and 5a show that the achieved recall decreases when the number of keywords increases from 2 keywords to 4 keywords. In addition, the recall of CORR-S falls deeper when increasing number of keywords in that range. However, we surprisedly found that the recall of 5-keyword queries is higher than the recall of 3-keyword and 4-keyword queries, while it is natural to expect that the recall should gradually fall when the number of keywords increases. After observing the results of tested 5-keyword queries, we found that the 5-keyword queries with AND semantics are quite selective, resulting in many of data sources with zero-score. Thus, the recall of those queries significantly increases. In contrast, figures 4b and 5b indicate the precision of both CORR-P and CORR-S gradually reduces when the number of query keywords increases from 2 keywords to 5 keywords. However, the precision of CORR-P is less affected by the increasing of query keywords, comparing with CORR-S. This is because CORR-S sometimes overestimates data sources which partly contains some keywords from a long keyword query.

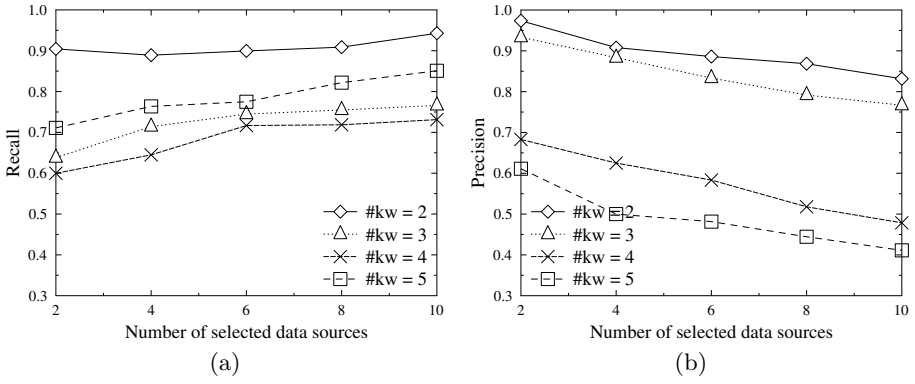


Fig. 4. Recall and precision of CORR-S w.r.t queries with various number of keywords

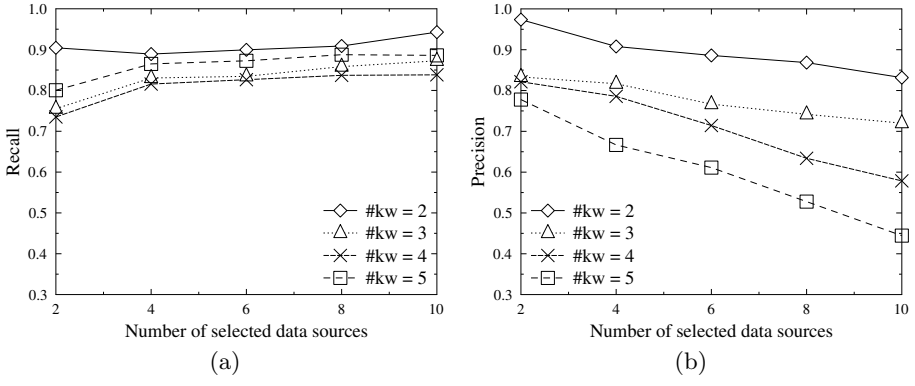


Fig. 5. Recall and precision of CORR-P w.r.t queries with various number of keywords

Effects of the number K of selected data sources. When the number of selected data sources increases from 1 to 20, the recall of all approaches increases at similar trend. This is because when the number of selected data sources increases, the chance of selecting relevant data sources also increases. Obviously, the recall of all approaches will get maximum value (equal to 1.0) when K is equal to the total number of data sources. However, when we select more data sources, the incidence of getting zero-score data sources can increase, which results in the decrease in the precision of all approaches.

6 Related Work

This section overviews some related work on keyword queries over single data source, query result ranking and some approaches on collection selection from the field of information retrieval.

6.1 XML Keyword Search over Single Data Source

Querying XML data using keyword-based search has been well studied in literature [3, 2, 1, 19, 6–12]. The baseline approach uses Lowest Common Ancestor (LCA) semantics from graph theory [20] to identify the result of a given keyword query. This approach returns a search result consisting of all candidates (i.e., subtrees) which each contains at least one instance of query keywords.

Recently, some variants of LCA semantics have been proposed [3, 9, 2, 19] to boost precision of the baseline approach. Amongst those approaches, the approach [9] was claimed to outperform the others, thus we use its semantics for retrieving "ideal" top-k data sources during our experiments.

Some other approaches studied the relationship between nodes containing query keywords to further filtering "meaningless" candidates. For instance, [2] defines that two match nodes u and v are meaningfully related if the shortest path between u and v does not have two distinct nodes with same label, except u and v. A candidate subtree is meaningful if it does not contain any pair of nodes which are not meaningful related to others. [19] proposes an approach to filter irrelevant candidates based on the dominance relationship between keyword nodes. Recently, [21] proposed an approach which prunes irrelevant subtrees based on the structural relationship between nodes containing the query keywords. Those proposed semantics can be used to define the meaningful relationship between keyword pairs for further reducing the size of our K-Graphs.

6.2 Query Result Ranking

Ranking XML keyword queries is an interesting problem that has been studied in the literature [3, 22]. The basic idea of ranking a keyword query result (e.g. a subtree) is that individual nodes directly containing the keywords can be viewed as "documents". Local ranking scores are given based on the "documents". An aggregation function aggregates them into a global score which is the final ranking score of the result subtree. Given a node v and a keyword w, $l(v, w)$ is a function that assigns to v a local ranking score. The function l can take multiple factors into account (e.g. IR score that evaluates the content relevance and link-based that evaluates the global importance of the node), and combine them in an arbitrary way. Note that any of those ranking approaches can be adaptably used for measuring the relationship between keywords in our K-Graph summaries.

In [3], the local score of a node v_i directly containing keyword w is calculated by adapting the PageRank, which is widely used to evaluate the importance of HTML documents. Each element in XML document is mapped to a document meanwhile edges (IDREF, XLink ad containment edges) are mapped to hyperlink edges. [22] proposed a novel ranking formulae to identify the search for nodes and search via nodes of a query, and present a novel XML TF∗IDF ranking strategy to rank the individual matches of all possible search intentions.

6.3 Collection Selection for Un-structured Data

Text collection selection has been widely studied in the field of Information Retrieval [23–25]. The main idea in existing systems is to try to create a representative for each collection based on term and document frequency information, and then use that information at query-time to determine which collections are most promising for the incoming query. However, as we pointed out in Section 1, applying those approaches in the context of XML data source selection is inefficient because it can ignore the structure of data.

7 Conclusions

In this paper, we have proposed an approach for selecting the top-k data sources to a given query in order to avoid high cost of search in numerous, potentially irrelevant data sources. The approach can efficiently select top-k mostly relevant data sources without querying over the data sources. A ranking function has been proposed for measuring the strength of keyword relationships in a data source and summarize the data sources as keywords correlation graphs (K-Graphs) to maintain the keyword relationships. The top-k relevant data sources are selected using our two ranking functions which can estimate the relevance of corresponding K-Graphs to the query. We conducted experiments using real data set and the results show that our approach achieves good performance in all evaluation metrics *recall*, *precision* in a variety of experimental parameters. We currently work on the issue of how to reduce the size of the summarized K-Graphs by (i) allow users to limit the allowed maximum distance between two keywords and (ii) extend the proposed semantics in literature to define the meaningful relationship between keywords.

References

1. Cohen, S., Kanza, Y., Kimelfeld, B., Sagiv, Y.: Interconnection semantics for keyword search in xml. In: Proceedings of CIKM, pp. 389–396. ACM, New York (2005)
2. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: Xsearch: a semantic search engine for xml. In: Proceedings of VLDB Endowment, pp. 45–56 (2003)
3. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: ranked keyword search over xml documents. In: Proceedings of SIGMOD, pp. 16–27. ACM, New York (2003)
4. Hristidis, V., Koudas, N., Papakonstantinou, Y., Srivastava, D.: Keyword Proximity Search in XML Trees. TKDE, 525–539 (2006)
5. Li, G., Feng, J., Wang, J., Zhou, L.: Effective keyword search for valuable lcas over xml documents. In: Proceedings of CIKM, pp. 31–40. ACM, New York (2007)
6. Liu, Z., Chen, Y.: Identifying meaningful return information for xml keyword search. In: Proceedings of SIGMOD, pp. 329–340. ACM, New York (2007)
7. Liu, Z., Walker, J., Chen, Y.: Xseek: a semantic xml search engine using keywords. In: Proceedings of VLDB Endowment, pp. 1330–1333 (2007)

8. Shao, F., Guo, L., Botev, C., Bhaskar, A., Chettiar, M., Yang, F., Shanmugasundaram, J.: Efficient keyword search over virtual xml views. In: Proceedings of VLDB Endowment, pp. 1057–1068 (2007)
9. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: Proceedings of SIGMOD, pp. 527–538. ACM, New York (2005)
10. Sun, C., Chan, C.Y., Goenka, A.K.: Multiway slca-based keyword search in xml data. In: Proceedings of WWW, pp. 1043–1052. ACM, New York (2007)
11. Xu, Y., Papakonstantinou, Y.: Efficient lca based keyword search in xml data. In: Proceedings of EDBT, pp. 535–546. ACM, New York (2008)
12. Zhou, R., Liu, C., Li, J.: Fast elca computation for keyword queries on xml data. In: Proceedings of EDBT, pp. 549–560. ACM, New York (2010)
13. Hadjieleftheriou, M., Chandel, A., Koudas, N., Srivastava, D.: Fast indexes and algorithms for set similarity selection queries. In: Proceeding of ICDE, pp. 267–276. IEEE Computer Society, Washington, DC, USA (2008)
14. http://dblp.unitrier.de/xml/
15. http://www.oracle.com/technology/products/berkeleydb/index.html
16. Chen, L.J., Papakonstantinou, Y.: Supporting top-k keyword search in xml databases. In: Proceeding of ICDE, pp. 689–700 (2010)
17. Powell, A.L., French, J.C.: Comparing the performance of collection selection algorithms. ACM Trans. Inf. Syst. 21, 412–456 (2003)
18. Gravano, L., García-Molina, H., Tomasic, A.: Gloss: text-source discovery over the internet. ACM Trans. Database Syst. 24, 229–264 (1999)
19. Li, G., Feng, J., Wang, J., Zhou, L.: Effective keyword search for valuable lcas over xml documents. In: Proceedings of CIKM, pp. 31–40. ACM, New York (2007)
20. Bender, M.A., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P.: Lowest common ancestors in trees and directed acyclic graphs. Journal of Algorithms 57, 75–94 (2005)
21. Li, Y., Yu, C., Jagadish, H.V.: Enabling schema-free xquery with meaningful query focus. The VLDB Journal 17(3), 355–377 (2008)
22. Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective xml keyword search with relevance oriented ranking. In: Proceedings of ICDE, Washington, DC, USA, pp. 517–528 (2009)
23. Gravano, L., García-Molina, H., Tomasic, A.: Gloss: text-source discovery over the internet. ACM Trans. Database Syst. 24, 229–264 (1999)
24. Yuwono, B., Lee, D.L.: Server ranking for distributed text retrieval systems on the internet. In: Proceedings of DASFAA, pp. 41–50. World Scientific Press, Singapore (1997)
25. Callan, J.P., Lu, Z., Croft, W.B.: Searching distributed collections with inference networks. In: Proceedings of SIGIR, pp. 21–28. ACM, New York (1995)

Detecting Economic Events Using a Semantics-Based Pipeline

Alexander Hogenboom, Frederik Hogenboom, Flavius Frasincar, Uzay Kaymak, Otto van der Meer, and Kim Schouten

Erasmus University Rotterdam
P.O. Box 1738, NL-3000 DR
Rotterdam, The Netherlands
{hogenboom,fhogenboom,frasincar,kaymak}@ese.eur.nl,
{276933rm,288054ks}@student.eur.nl

Abstract. In today's information-driven global economy, breaking news on economic events such as acquisitions and stock splits has a substantial impact on the financial markets. Therefore, it is important to be able to automatically identify events in news items accurately and in a timely manner. For this purpose, one has to be able to mine a wide variety of heterogeneous sources of unstructured data to extract knowledge that is useful for guiding decision making processes. We propose a Semantics-based Pipeline for Economic Event Detection (SPEED), which aims at extracting financial events from news articles and annotating these events with meta-data, while retaining a speed that is high enough to make real-time use possible. In our pipeline implementation, we have reused some of the components of an existing framework and developed new ones, such as an Ontology Gazetteer and a Word Sense Disambiguator.

1 Introduction

In today's information-driven society, machines that can perform Natural Language Processing (NLP) tasks can be of great importance. Decision makers are expected to process a continuous, overwhelming flow of (news) messages by extracting information and understanding their meaning. Knowledge can subsequently be acquired by applying reasoning to the gathered information. In today's global economy, it is of paramount importance for decision makers to have a sensible intuition on the state of their market, which is often extremely sensitive to breaking news on economic events like acquisitions, stock splits, dividend announcements, etc. In this context, identification of events can guide decision making processes, as these events provide means of structuring information using concepts, with which knowledge can be generated by applying inference. Automating information extraction and knowledge acquisition processes can facilitate or support decision makers in fulfilling their cumbersome tasks, as one can make better informed decisions due to faster processing of more data.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 440–447, 2011.

Therefore, we aim to have a fully automated framework for processing financial news messages gathered from Really Simple Syndication (RSS) feeds. These events are represented in a machine-understandable way. Extracted events can be made accessible for other applications through the use of Semantic Web technologies. Furthermore, we aim for the framework to be able to handle news messages at a speed useful for real-time use, as new events can occur any time and require decision makers to respond in a timely and adequate manner.

We propose a pipeline that identifies the concepts related to economic events, which are defined in a domain ontology and are associated to synsets from a semantic lexicon such as WordNet [3]. For concept identification, we employ lexico-semantic patterns based on ontology concepts in order to match lexical representations of concepts retrieved from the text with event-related concepts that are available in the semantic lexicon, and thus aim to maximize recall. The identified lexical representations of relevant concepts are subject to a Word Sense Disambiguation (WSD) procedure for determining the corresponding sense, in order to maximize precision. To enable real-time use, we also aim to minimize the latency, i.e., the time it takes for the pipeline to process a news message.

The remainder of this paper is structured as follows. First, Sect. 2 discusses related work. Subsequently, Sect. 3 elaborates on the proposed framework. Then, the framework is evaluated in Sect. 4, after which Sect. 5 concludes the paper.

2 Related Work

Several tools have already been proposed for our desired Information Extraction (IE) purposes, most of which have their own IE framework. However, the General Architecture for Text Engineering (GATE) [2], a freely available general purpose framework for IE purposes, has become increasingly popular. The tool is highly flexible in that the user can construct processing pipelines from components that perform specific tasks. One can distinguish between linguistic analysis applications such as tokenization (e.g., distinguishing words), syntactic analysis jobs like Part-Of-Speech (POS) tagging, and semantic analysis tasks such as understanding. By default, GATE loads the A Nearly-New Information Extraction (ANNIE) system, which consists of several key components, i.e., the *English Tokenizer*, *Sentence Splitter*, *Part-Of-Speech (POS) Tagger*, *Gazetteer*, *Named Entity (NE) Transducer*, and *OrthoMatcher*.

Although the ANNIE pipeline has proven to be useful in various information extraction jobs, its functionality does not suffice when applied to discovering economic events in news messages. For instance, ANNIE lacks important features such as a WSD component, although some disambiguation can be done using JAPE rules in the *NE Transducer*. This is however a cumbersome and ineffective approach where rules have to be created manually for each term, which is prone to errors. Furthermore, ANNIE lacks the ability to individually look up concepts from a large ontology within a limited amount of time. Nevertheless, GATE is highly flexible and customizable, and therefore ANNIE's components are either usable, extendible, or replaceable in order to suit our needs.

An example of a tool utilizing ANNIE components is Hermes [4], which extracts a set of news items related to specific concepts of interest. ANNIE components are used that make use of concepts and relations stored in ontologies. Another example of an adapted ANNIE pipeline is the Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations (CAFETIERE) relation extraction pipeline [1], which contains an ontology look-up process and a rule engine. CAFETIERE employs extraction rules defined at lexico-semantic level which are more easy to express, yet less flexible than JAPE rules. CAFETIERE stores knowledge in a type of ontology which has no formal semantics and lacks reasoning support, rendering this an unattractive approach for identifying, e.g., financial events. Furthermore, gazetteering is a slow process when going through large ontologies. Finally, the pipeline also misses a WSD component.

The Knowledge and Information Management (KIM) platform [8] provides an infrastructure for IE purposes, by combining the GATE architecture with semantic annotation techniques. KIM focuses on automatic annotation of news articles, where entities, inter-entity relations, and attributes are discovered. For this, a pre-populated Web Ontology Language (OWL) upper ontology is employed, i.e., a minimal but sufficient ontology that is suitable for open domain and general purpose annotation tasks. The semantic annotations in articles allow for applications such as semantic querying and exploring the semantic repository. The differences between KIM and our approach are in that we aim for a financial event-focused information extraction pipeline, which is in contrast to KIM's general purpose framework. Hence, we employ a domain-specific ontology instead of an upper ontology. Also, rather than just annotating corpora with event concepts, we extract additional information by utilizing lexico-semantic patterns for linking identified concepts, thus realizing a rich knowledge base. Furthermore, no mention has been made regarding WSD within the KIM platform, whereas we consider WSD to be an essential component in an IE pipeline.

3 Economic Event Detection Based on Semantics

Where current approaches to automated IE from news messages are more focused on annotation of documents, we strive to actually extract information – i.e., specific economic events – from documents, with which for instance a knowledge base can be updated. The analysis of texts needs to be driven by semantics, as the domain-specific information captured in these semantics facilitates detection of relevant concepts. Therefore, we propose a Semantics-Based Pipeline for Economic Event Detection (SPEED), consisting of several components which sequentially process documents. This approach is driven by an ontology containing information on the NASDAQ-100 companies, extracted from Yahoo! Finance. This domain ontology has been developed by domain experts through an incremental middle-out approach, validated using the OntoClean methodology [5]. The ontology captures concepts and events from the financial domain, e.g., companies, competitors, products, etc. Many concepts in this ontology stem from a semantic lexicon (e.g., WordNet) or represent named entities.

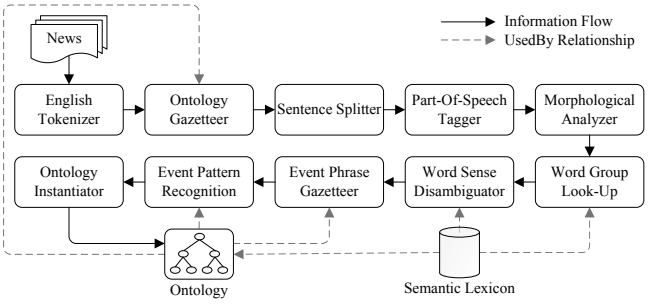


Fig. 1. SPEED design

Our proposed pipeline, depicted in Fig. 1, is designed to identify relevant concepts and their relations in a document. To this end, individual components of the text are first identified as such by means of the *English Tokenizer*, which splits text into tokens (e.g., words or numbers) while taking into account rules specific to the English language. These tokens are then linked to ontology concepts by an *Ontology Gazetteer*. Matching tokens in the text are thus annotated with a reference to their associated concepts defined in the ontology.

Then, the *Sentence Splitter* groups the tokens in the text into sentences, based on tokens indicating a separation between sentences, e.g., (a combination of) punctuation symbols or new line characters. These sentences are used for discovering the grammatical structure in text by determining the part-of-speech of each word token by means of the *Part-Of-Speech Tagger*. As words can have many forms that have a similar meaning, the *Morphological Analyzer* subsequently reduces the tagged words to their lemma and an affix.

Words and meanings, denoted often as synsets (set of synonyms) have a many-to-many relationship. Hence, the next step in interpreting a text is disambiguation of its words' meaning, given their POS tags, lemmas, etc. To this end, a *Word Group Look-Up* component first combines words into word groups containing as many words per group as possible for representing some concept in the semantic lexicon. The *Word Sense Disambiguator* then determines the word sense of each word group by exploring the mutual relations between senses (as defined in the semantic lexicon and the ontology) of word groups; the stronger the relation with surrounding senses, the more likely a sense matches the context.

To this end, we propose an adaptation of the Structural Semantic Interconnections (SSI) [7] algorithm. The SSI approach uses graphs to describe word groups and their context (word senses), as derived from a semantic lexicon. The senses are determined based on the number and type of detected semantic interconnections in a labeled directed graph representation of all senses of the considered word groups. We differ from SSI in that our algorithm, shown in Algorithm 1, considers the two most likely senses for each word group and iteratively disambiguates the word group with the highest confidence (i.e., weighted difference of the similarity of both senses to already disambiguated senses), rather than the word group with the greatest similarity for its best sense. Furthermore, in case

```
a = ∅; // Lists ambiguous word groups yet to be disambiguated
d = ∅; // Lists disambiguated word groups
s = ∅; // Lists senses of disambiguated word groups
c = ∅; // Lists context (i.e., possible senses of all considered word groups)
l = ∅; // Lists similarity of context to disambiguated senses
// Initialize disambiguation
w = getWordGroups();
foreach g in w do
    senses = getSenses(g);
    // Add word group g with one sense to d and its sense to s
    if |senses| == 1 then
        add(d,g);
        add(s,senses);
    // Add ambiguous word group g to a and its senses to c
    else
        add(a,g);
        foreach sense in senses do if sense ∉ c then add(c,sense);
    end
end
// Determine similarity of all senses in c to all disambiguated senses in s
foreach sense in c do
    simToS = 0;
    foreach knownSense in s do simToS += 1/shortestPathLength(sense,knownSense);
    add(l,simToS);
end
// Disambiguate word groups in a
lastAddedSense = ∅;
while a ≠ ∅ do
    bestPick,bestPickSense = ∅;
    bestPickConf = −∞;
    foreach g in a do
        bestSense1,bestSense2 = ∅;
        bestSim1,bestSim2 = −∞;
        senses = getSenses(g);
        foreach sense in senses do
            // Update similarity of sense to s with similarity to lastAddedSense
            indexSense = indexOf(c,sense);
            simToS = get(l,indexSense);
            simToS += 1/shortestPathLength(sense,lastAddedSense);
            set(l,indexSense,simToS);
            // Update best senses
            if simToS > bestSim2 then
                if simToS > bestSim1 then
                    bestSense2 = bestSense1; bestSense1 = sense;
                    bestSim2 = bestSim1; bestSim1 = simToS;
                else
                    bestSense2 = sense;
                    bestSim2 = simToS;
                end
            end
        end
        // Update best pick
        confidence = ((bestSim1-bestSim2)*bestSim1);
        if confidence > bestPickConf then
            bestPick = g;
            bestPickSense = bestSense1;
            bestPickConf = confidence;
        end
    end
    // Disambiguate best pick, move it from a to d, and add its sense to s
    rem(a,indexOf(a,bestPick));
    add(d,bestPick);
    add(s,bestPickSense);
    lastAddedSense = bestPickSense;
end
```

Algorithm 1. Word Sense Disambiguation for an arbitrary news item

an arbitrary word cannot be disambiguated, we default to the statistically most likely sense in our semantic lexicon, whereas the original SSI algorithm fails to provide a word sense. In our algorithm, we compute the similarity of a sense to already disambiguated senses as the sum of the inverse of the shortest path length between this sense and the disambiguated senses in the semantic graph.

When the meaning of word groups has been disambiguated, the text can be interpreted by introducing semantics linking word groups to an ontology, thus capturing their essence in a meaningful and machine-understandable way. As we are interested in specific economic events, the *Event Phrase Gazetteer* scans the text for those events. It uses a list of phrases or concepts that are likely to represent some part of a relevant event. Events thus identified are then supplied with available additional information (e.g., time stamps) by the *Event Pattern Recognition* component, which matches identified events with predefined domain-specific lexico-semantic patterns. Finally, the knowledge base can be updated by inserting the identified events and their extracted associated information into the ontology using the *Ontology Instantiator*, as detailed in our previous work [9].

4 Evaluation

The modularity of an architecture like GATE can facilitate the implementation and subsequent evaluation of our proposed semantics-based pipeline for economic event detection. Therefore, we have made a Java-based implementation of the proposed framework, partially using default GATE components which suit our needs, i.e., the *English Tokenizer*, *Sentence Splitter*, *Part-Of-Speech Tagger*, and the *Morphological Analyzer*. Additionally, we have extended the functionality of other GATE components (e.g., for ontology gazetteering), and also implemented new components to tackle the WSD and economic event detection processes.

The implementations of both our *Ontology Gazetteer* and *Word Group Look-Up* components match concepts (i.e., ontology concepts and WordNet word groups, respectively) with lexical representations stored in a look-up tree, where nodes represent individual tokens and a path from the root node to an arbitrary leaf node represents a concept's lexical representation. For each token, the look-up tree is consulted, starting from the root node. If the token is not in the root, the next token in the text is again looked up in the root. Else, the next token in the text is looked up in the root node of the subtree belonging to the former token. This process is iterated until either a leaf node is reached, or the current node does not have a reference to the next token in the text. The word group associated with the followed path is then annotated with the associated concept. Our trees for ontology concepts and word groups have been implemented using hash maps, in order to reduce the time needed to traverse the trees.

In order to evaluate SPEED's performance, we assess statistics that describe the cumulative error, i.e., precision and recall, and latency. We define precision as the part of the identified concepts (e.g., word senses or events) that have been identified correctly, and recall represents the number of identified concepts as a fraction of the number of concepts that should have been identified. When

we compare the performance of different approaches, we assess the statistical relevance of differences in performance by means of a paired t-test.

We evaluate our *Word Sense Disambiguator* on a large, publicly available WSD corpus – SemCor [6]. On this corpus, the original SSI algorithm exhibits an average precision of 53% with a standard deviation of 5 percentage points and a recall of 31% with a standard deviation of 9 percentage points. Conversely, our proposed adaptation of SSI exhibits an average precision and recall of 59% with a standard deviation of 5 percentage points. This implies an overall improvement in precision and recall with 12% and 90%, respectively, compared to the original SSI algorithm, at a significance level of 0.001.

In the evaluation of our framework as a whole, we focus on a data set consisting of 200 news messages extracted from the Yahoo! Business and Technology newsfeeds. Three domain experts have manually annotated these for our considered economic events and relations, while ensuring an inter-annotator agreement of at least 66% (i.e., at least two out of three annotators agree). We distinguish between ten different financial events, i.e., announcements regarding CEOs (60), presidents (22), products (136), competitors (50), partners (23), subsidiaries (46), share values (45), revenues (22), profits (33), and losses (27).

We observe a precision for the concept identification in news items of 86% and a recall of 81%. It should however be noted that precision and recall of fully decorated events result in lower values of approximately 62% and 53% respectively, as they rely on multiple concepts that have to be identified correctly. Errors in concept identification result from missing lexical representations of the knowledge base concepts, and missing concepts in general. Despite using only WordNet as a semantic lexicon, we obtain high precision as many of our concepts' lexical representations are named entities, which often are monosemous. High recall can be explained by SPEED's focus on detecting ontology concepts in the text, rather than on identifying all concepts in the text.

On our data set, our pipeline exhibits a latency of on average 632 milliseconds per document, with a standard deviation of 398 milliseconds. Of this execution time, roughly 30% is allocated to the first part of the pipeline, performing linguistic and syntactic analysis tasks. The subsequent WSD task on average takes up about 60% of the execution time, whereas the remaining tasks are typically performed in about 10% of the execution time.

5 Conclusions and Future Work

We have proposed a semantics-based framework for economic event detection (SPEED), which aims to extract financial events from news articles (announced through RSS feeds) and to annotate these with meta-data, while maintaining a speed that is high enough to enable real-time use. For implementing the SPEED pipeline, we have reused existing components and developed new ones such as gazetteers and word sense disambiguator. Our framework is semantically enabled, i.e., it makes use of semantic lexicons and ontologies. Furthermore, pipeline outputs also make use of semantics, which introduces a potential feedback loop, making event identification a more adaptive process. The merit of our

pipeline is in the use of semantics, enabling broader application interoperability. Although we focus on the financial domain, SPEED is generalizable to other domains, as we separate the domain-specific aspects from the domain-independent ones. The established fast processing time and high precision and recall provide a good basis for future work.

For future work, we aim to investigate further possibilities for implementation in algorithmic trading environments. We aim to find a way of utilizing discovered events in this field. To this end, we also envision another addition, i.e., a way of associating sentiment with discovered events. As sentiment of actors with respect to events may be the driving force behind their reactions to these events, this information could be exploited in an algorithmic trading setup.

Acknowledgment. The authors are partially sponsored by the NWO Physical Sciences Free Competition project 612.001.009: Financial Events Recognition in News for Algorithmic Trading (FERNAT).

References

[1] Black, W.J., M^{c}Naught, J., Vasilakopoulos, A., Zervanou, K., Theodoulidis, B., Rinaldi, F.: CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations. Technical Report TR–U4.3.1. Department of Computation, UMIST, Manchester (2005)

[2] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: 40th Anniversary Meeting of the ACL (ACL 2002), pp. 168–175. ACL (2002)

[3] Fellbaum, C.: WordNet an Electronic Lexical Database. Computational Linguistics 25(2), 292–296 (1998)

[4] Frasincar, F., Borsje, J., Levering, L.: A Semantic Web-Based Approach for Building Personalized News Services. International Journal of E-Business Research 5(3), 35–53 (2009)

[5] Guarino, N., Welty, C.A.: Evaluating Ontological Decisions with OntoClean. Communications of the ACM 45(2), 61–65 (2002)

[6] Miller, G., Chodorow, M., Landes, S., Leacock, C., Thomas, R.: Using a Semantic Concordance for Sense Identification. In: Proceedings of the Human Language Technology Workshop (HLT 1994), pp. 240–243. ACL (1994)

[7] Navigli, R., Velardi, P.: Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. Trans. Pattern Anal. Machine Intell. 27(7), 1075–1086 (2005)

[8] Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: KIM - A Semantic Platform For Information Extraction and Retrieval. Journal of Natural Language Engineering 10(3-4), 375–392 (2004)

[9] Schouten, K., Ruijgrok, P., Borsje, J., Frasincar, F., Levering, L., Hogenboom, F.: A Semantic Web-Based Approach for Personalizing News. In: 25th Symposium On Applied Computing (SAC 2010), pp. 854–861. ACM, New York (2010)

Edit Distance between XML and Probabilistic XML Documents

Ruiming Tang[1], Huayu Wu[1], Sadegh Nobari[1], and Stéphane Bressan[2]

[1] School of Computing, National University of Singapore
{tangruiming,wuhuayu,snobari}@comp.nus.edu.sg
[2] Center for Maritime Studies
steph@nus.edu.sg

Abstract. Probabilistic XML is a hierarchical data model capturing uncertainty of both value and structure. The ability to compute the similarity between an XML document and a probabilistic XML document is a building block of many applications involving querying, comparison, alignment and classification, for instance. The new challenge in efficiently computing such similarity is the multiplicity of the possible worlds represented by a probabilistic XML document. We devise and discuss an algorithm for the efficient computation of the similarity between an XML document and a probabilistic XML document. We empirically and comparatively evaluate the performance of the algorithm and its variants.

1 Introduction

1.1 Motivation

Modern applications (see [13] and [16], for instance) wanting to exploit the now available numerous data sources face the challenge posed by uncertainty of information. Unfortunately there is little room for uncertainty in traditional database models and management systems. New models, new tools and techniques are needed (see [4]).

Probabilistic XML is a hierarchical data model capturing uncertainty of both value and structure. The computation of similarity is an essential building block for the tasks at hand, namely comparison, alignment, clustering and classification of data. Several algorithms exist for measuring the structural similarity between XML documents among themselves or XML documents and XML document type definitions and schemas. The new challenge in efficiently computing the similarity between an XML document and a probabilistic XML document is the multiplicity of the possible worlds that a probabilistic XML document represents.

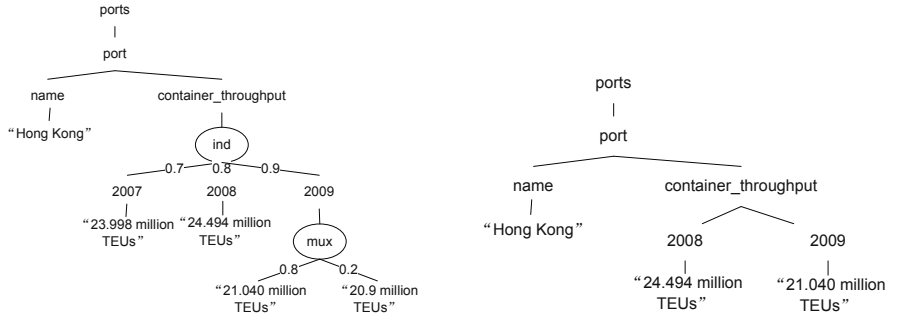
In this paper, we devise and discuss an algorithm and its variants for computing the similarity between an XML document and a probabilistic XML document. The algorithms implement the expected value of the edit distance between an XML document and the documents in the set defined by the probabilistic XML document. We empirically and comparatively evaluate their performance.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 448–456, 2011.

In the absence of established corpora and benchmarks for probabilistic XML, we also propose and use several random probabilistic XML models together with the associated random generation algorithms.

1.2 Probabilistic XML

Fig. 1 shows a probabilistic XML document in the model of [13]. A probabilistic XML document is an XML document with special nodes. In the figure the special nodes are labeled "mux" and "ind". These nodes are called *distributional nodes*.



(a) one probabilistic XML document (b) one possible world

Fig. 1. A Probabilistic XML Document and one of its Possible World

A distributional node labeled "mux" denotes a *mutually exclusive* choice among its children (or none of the children if the sum of probabilities is less than one) with a probability associated to each child. A distributional node labeled "ind" denotes an *independent* choice among its children with a probability associated to each child. There are also models with two additional distributional nodes, i.e., "exp" and "cie", as reviewed in Section 2. In our work, we only consider probabilistic XML documents with mutually exclusive and independent distributional nodes.

In general, a probabilistic XML document U is a random model of XML documents. It is a probability space that defines an assignment of probabilities to all ordinary XML documents or, in other words, a probability distribution over XML documents. Each XML document with a non zero probability, $Pr(d) > 0$, is called a *possible world* of the probabilistic document U. We say that a possible world d, has a probability $Pr(d)$ to belong to $pwd(U)$, representing the set of possible worlds defined by the probabilistic XML document U. Clearly $\sum_{d \in pwd(U)} Pr(d) = 1$.

1.3 Organization

In the following, related work is synthesized in Section 2. We define similarity and present the algorithms for its computation in Section 3. Section 4 presents

and discusses the results of the evaluation of our algorithms. It also discusses the generation of synthetic probabilistic XML data. Finally, we conclude in Section 5.

2 Related Work

2.1 Probabilistic XML Models

[1] surveys the different probabilistic XML data models that have been proposed in the literature. These models differ in the distributional nodes that they consider: mutually exclusive choices, independent choices, explicit choices and conjunction of independent events. Following [1], we refer to these models according to the distributional nodes that they consider. The mostly used models are $PrXML^{mux}$, $PrXML^{mux,ind}$, $PrXML^{exp}$ and $PrXML^{cie}$, respectively.

The authors of [13] introduce the $PrXML^{mux}$ and $PrXML^{mux,ind}$ models, the latter being more expressive than the former. They propose query processing techniques for these models. The authors of [2] introduce the notion of independent events and the $PrXML^{cie}$ model. They show the added expressiveness of $PrXML^{cie}$ over $PrXML^{mux,ind}$. They define query evaluation and update techniques for both $PrXML^{mux,ind}$ and $PrXML^{cie}$. The authors of [7] introduce the $PrXML^{exp}$ model together with semantics, algebra and efficient algorithms for query processing. The model proposed in [16] is similar to $PrXML^{mux,ind}$. It is used to integrate disparate data. The authors of [9] study the evaluation of twig queries for probabilistic XML in the $PrXML^{mux,ind}$ model. They broaden their study to $PrXML^{mux}$, $PrXML^{mux,ind}$, $PrXML^{exp}$, and $PrXML^{cie}$ in [8].

There seems to be a natural tradeoff between the expressiveness of a model and the efficient of query evaluation. $PrXML^{mux}$ is the least expressive model while the most expressive are $PrXML^{cie}$ and $PrXML^{exp}$. Their combination $PrXML^{cie,exp}$ subsumes all the other models. We believe that $PrXML^{mux,ind}$ strikes a good balance between the expressiveness and efficiency. This model is also the most commonly adapted.

2.2 Similarity

Similarity metrics and distances attempt to quantify the resemblance of different objects. The edit distance, introduced for strings in [11] and [17], quantifies this resemblance as the minimum cost of transforming one object into another. The lower the cost is, the more similar the objects.

The idea of a tree edit distance is introduced by Tai in [14] in 1979. Zhang and Shasha [18] propose definition and algorithm for edit distance of trees. In this paper we consider the tree edit distance proposed in [5] that allows update of any vertex in the tree but restricts deletions and insertions to leaves. Different similarity definitions and the algorithms to effectively and efficiently compute them have been used for data clustering [12], change detection [6], schema integration [10], etc.

3 Proposal

3.1 Edit Distance

The edit distance between two trees is the minimum cost of the operations needed to transform one tree into another. In this paper we consider the definition of [5] in which three unit cost operations are considered: deletion of a leaf, insertion of a leaf and update of a node anywhere in the tree. We refer to this edit distance between two XML document as δ_d. δ_d can be relatively efficiently computed using dynamic programming.

A probabilistic XML document is a probability space. The edit distance between an XML document d and a probabilistic XML document U can be defined as the minimum, maximum or average distance between the XML document and the possible worlds. In this paper we choose to define it as the expected value of the edit distance between the XML document and the possible worlds. We refer to this edit distance as δ_p. It is the sum of the tree edit distances between the XML document and the possible worlds weighted with probabilities, as given in Equation 1.

$$\delta_p(d, U) = \sum_{w \in pwd(U)} Pr(w) \times \delta_d(d, w) \tag{1}$$

In this paper we use and adapt the tree edit distance and the algorithm of [5]. The technique and the data structure, called the "edit graph" in [5], is itself an extension to trees to the original dynamic programming approach and the matrix data structure used in [17] for string edit distance.

Next we present the stack algorithm to compute the edit distance between an XML document and a probabilistic XML document[1]. In order to illustrate it, we first present the naive enumeration algorithm and the multidimensional algorithm.

3.2 Enumeration Algorithm

The enumeration algorithm, E, generates the possible worlds for the probabilistic XML document U with their respective probabilities. Then, it computes the edit distance between the XML document and each of the possible world. Finally it sums the products of the distances with the respective probabilities. In this algorithm, all the pairwise distances between the XML document and the possible worlds are needed.

3.3 Multidimensional Algorithm

The main idea for an efficient edit distance algorithm for probabilistic XML is to share common computations. Indeed, while distributional nodes correspond to branching to possible worlds, all other nodes in a probabilistic XML are normal

[1] Due to the space limitation, the pseudo-codes of the algorithms are omitted in this paper. They are available from our technical report [15].

XML nodes and can be processed as such. The algorithm, that we call the "multidimensional algorithm" or MA for short, copies the dynamic programming sub matrices when it encounters distributional nodes. It branches the computation to the possible worlds or dimensions. This allows the sharing of computation until the next distributional node. The algorithm is further improved by noticing that the dynamic programming approach does not require to memorizing the entire matrix but rather one column at a time.

3.4 Stack Algorithm

Fig. 2 illustrates the successive data structures needed in the computation of the edit distance between an XML document and a probabilistic XML document.

The figure immediately suggests reconsidering the strategy of the MA algorithm. Instead of memorizing the possible worlds and copying the columns of the dynamic programming matrix we can compute the distance by a depth first strategy that only requires remembering the data structures at the branching points in the tree. This is classically done with a stack. Based on this idea, we improve the MA algorithm to be the MAS algorithm.

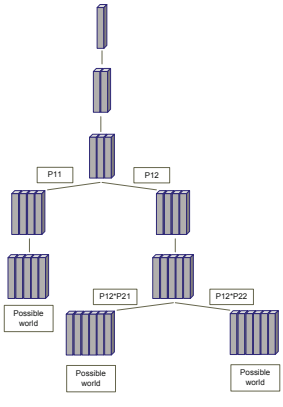


Fig. 2. MA and MAS Data Structure

4 Performance Evaluation

4.1 Experimental Setup

The three algorithms are implemented in Java. All experiments are run on a Centos 5.5 2 x Quad-Core Xeon E5520 2.2GHz with 24.0GB RAM.

The experiments are conducted with synthetic data. In order to control the generation of synthetic data we propose three random models for probabilistic XML: $P_X(doc, n)$, $P_X(doc, p)$ and $P_X(doc, p, f)$. These random probabilistic XML models are probability spaces of XML documents.

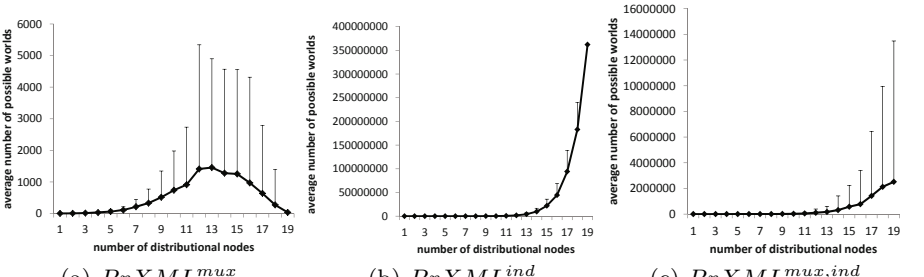
A probabilistic XML document in the $P_X(doc, n)$ model is derived from the original XML document *doc* by inserting independently n internal distributional nodes. A probabilistic XML document in the $P_X(doc, p)$ model is derived from the original XML document *doc* by inserting independently distributional nodes below each internal node with probability p. Finally, in order to control the depth at which distributional nodes appear, we propose the $P_X(doc, p, f)$ model in which a probabilistic XML document is derived from the original XML document *doc* by inserting independently distributional nodes below each internal node with probability $p \times f(q)$, where q is the level of the node in the XML tree. For each of these models we devise an algorithm that generates uniformly at random a probabilistic XML document in the model.

We use these three models and the corresponding algorithms for our experiments. In the experiments *doc* is a synthetic XML document generated using ToXGene [3]. It has 43 nodes, 19 internal nodes, a maximum depth of 4, an average depth of 2.558, a maximum number of children for the internal nodes of 4, and an average number of children of 2.211.

4.2 Experiments

Possible Worlds. In this first experiment we measure the number of possible worlds as a function of the distributional nodes. We consider the three $PrXML^{mux}$, $PrXML^{ind}$ and $PrXML^{mux,ind}$ models.

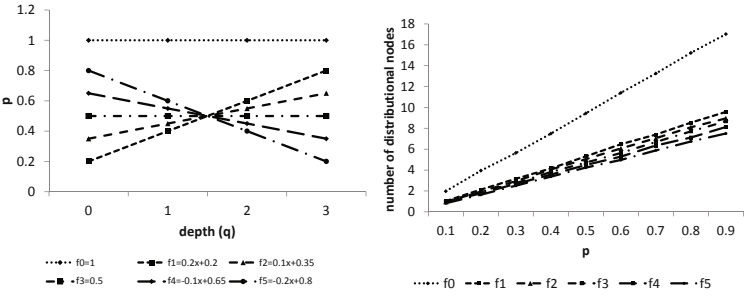
We generate 1000 probabilistic XML documents (there may not be so many different possible probabilistic XML documents, however by generating so many we get a better uniformity of the sample) in the $P_X(doc, n)$ model for n varying from 1 to 19. We plot the average value and its standard deviation.



(a) $PrXML^{mux}$ (b) $PrXML^{ind}$ (c) $PrXML^{mux,ind}$

Fig. 3. Number of possible worlds as a function of the distributional nodes

Fig. 3.a shows that, in the case of $PrXML^{mux}$, the number of possible worlds increases until it reaches a maximum ($n = 13$) and decreases again. This is because as the number of distributional nodes increases beyond 13 these modes are more likely to be nested and generate less possible worlds. Fig. 3.b shows that, in the case of $PrXML^{ind}$, the number of possible worlds increases. There is no peak. This is because even if distributional nodes of type "ind" are nested the



(a) six different functions

(b) number of distributional nodes against p with different functions

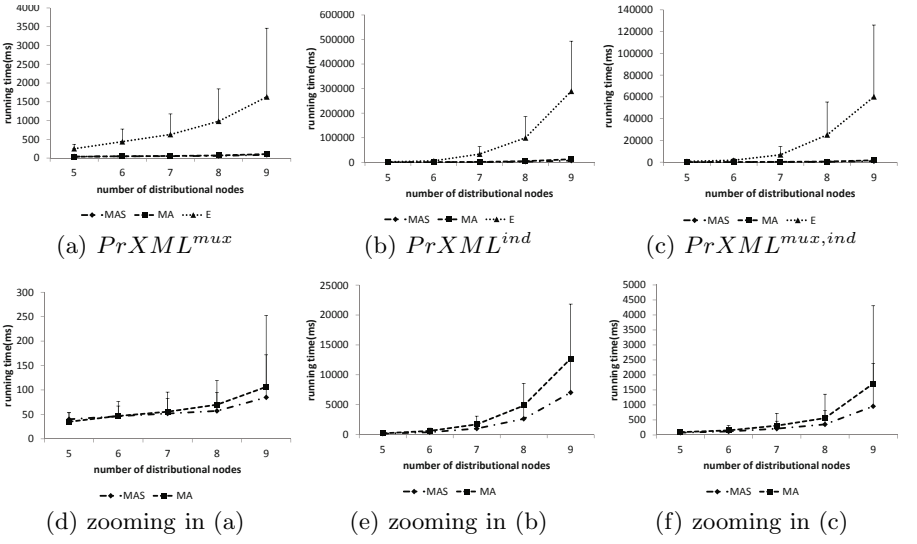
Fig. 4. Number of distributional nodes as a function of the probability

number of possible worlds that they generate is still combinatorial. Fig. 3.c shows that, in the case of $PrXML^{ind,mux}$, the number of possible worlds increases. There is no peak. This is because the effect of distributional nodes of type "ind" dominates the effect of distributional nodes of type "mux".

Number of Distributional Nodes. In this second experiment we measure the number of distributional nodes as a function of the probability of the nodes to be inserted at certain levels. We generate 1000 probabilistic XML documents in the $P_X(doc, p, f)$ model for p varying from 0.1 to 0.9. We plot the average value for the six functions f_0, f_1, f_2, f_3, f_4 and f_5 of Fig. 4.a. Notice that in the cases of f being a constant function ($f_0(q) = 1$ and $f_3(q) = 0.5$) $P_X(doc, p, f)$ is $P_X(doc, p \times f)$ and, therefore is in the $P_X(doc, p)$ model. Fig. 4.b shows that for the same compound probability (sum of the probabilities for each level: f_1 to f_5) the larger the slope the more distributional nodes. The case of f_0 also shows that the larger the probabilities the more distributional nodes.

Running Time. In this third experiment we measure the running time as a function of the number of distributional nodes. We consider the three $PrXML^{mux}$, $PrXML^{ind}$ and $PrXML^{mux,ind}$ models. We generate 1000 probabilistic XML documents in the $P_X(doc, n)$ for n varying from 5 to 9. We plot the average value and its standard deviation for the three algorithms: enumeration algorithm, E, multidimensional algorithm, MA, and the stack algorithm, MAS.

The average running time is increasing with the number of distributional nodes. On Fig. 5.a,b,c we see that the MA and MAS are significantly faster than E. On Fig. 5.d,e,f, zooming in the performance of MA and MAS, we see that the stack algorithm is faster. Although MA and MAS are just a breadth-first and depth-first traversal of the same tree, respectively, there is overhead in copying and memorizing the data structures in MA as opposed to pushing and popping the strictly necessary ones to and from the stack in MAS. MAS is the most efficient algorithm for computing the edit distance between an XML document and a probabilistic XML document.



(a) $PrXML^{mux}$ (b) $PrXML^{ind}$ (c) $PrXML^{mux,ind}$

(d) zooming in (a) (e) zooming in (b) (f) zooming in (c)

Fig. 5. Running time as a function of the number of distributional nodes

5 Conclusion

We have defined the similarity between an XML and a probabilistic XML document as the expected value of the edit distance between the XML document and the set of XML documents denoted by the probabilistic XML document. We have devised an original dynamic programming algorithm, MAS, and empirically shown that it outperforms the baseline approaches. In the absence of established corpora and benchmarks for probabilistic XML, we have also proposed several random probabilistic XML models and their corresponding random generation algorithms for the generation of synthetic probabilistic XML data.

References

1. Abiteboul, S., Kimelfeld, B., Sagiv, Y., Senellart, P.: On the Expressiveness of Probabilistic XML models. VLDB J. 18(5), 1041–1064 (2009)
2. Abiteboul, S., Senellart, P.: Querying and updating probabilistic information in XML. 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. 1059–1068. Springer, Heidelberg (2006)
3. Barbosa, D., Mendelzon, A.O., Keenleyside, J., Lyons, K.A.: ToXgene: An Extensible Template-Based Data Generator for XML. In: WebDB, pp. 49–54 (2002)
4. Cavallo, R., Pittarelli, M.: The Theory of Probabilistic Databases. In: VLDB, pp. 71–81 (1987)
5. Chawathe, S.S.: Comparing Hierarchical Data in External Memory. In: VLDB, pp. 90–101 (1999)

6. Cobena, G., Abiteboul, S., Marian, A.: Detecting Changes in XML Documents. In: ICDE, pp. 41–52 (2002)
7. Hung, E., Getoor, L., Subrahmanian, V.S.: PXML: A Probabilistic Semistructured Data Model and Algebra. In: ICDE, pp. 467–479 (2003)
8. Kimelfeld, B., Kosharovsky, Y., Sagiv, Y.: Query Efficiency in Probabilistic XML models. In: SIGMOD Conference, pp. 701–714 (2008)
9. Kimelfeld, B., Sagiv, Y.: Matching Twigs in Probabilistic XML. In: VLDB, pp. 27–38 (2007)
10. Leonardi, E., Hoai, T.T., Bhowmick, S.S., Madria, S.K.: DTD-Diff: A Change Detection Algorithm for DTDs. Data Knowl. Eng. 61(2), 384–402 (2007)
11. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10, 707 (1966)
12. Nierman, A., Jagadish, H.V.: Evaluating Structural Similarity in XML Documents. In: WebDB, pp. 61–66 (2002)
13. Nierman, A., Jagadish, H.V.: ProTDB: Probabilistic Data in XML. In: VLDB, pp. 646–657 (2002)
14. Tai, K.-C.: The Tree-to-Tree Correction Problem. J. ACM 26(3), 422–433 (1979)
15. Tang, R., Wu, H., Nobari, S., Bressan, S.: Technical Report: Edit Distance between XML and Probabilistic XML Documents, from http://www.comp.nus.edu.sg/~tang1987
16. van Keulen, M., de Keijzer, A., Alink, W.: A Probabilistic XML Approach to Data Integration. In: ICDE, pp. 459–470 (2005)
17. Wagner, R.A., Fischer, M.J.: The String-to-String Correction Problem. J. ACM 21(1), 168–173 (1974)
18. Zhang, K., Shasha, D.: Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. SIAM J. Comput. 18(6), 1245–1262 (1989)

Towards an Automatic Characterization of Criteria

Benjamin Duthil[1], François Trousset[1], Mathieu Roche[2], Gérard Dray[1], Michel Plantié[1], Jacky Montmain[1], and Pascal Poncelet[2]

[1] EMA-LGI2P, Parc Scientifique Georges Besse, 30035 Nîmes Cedex, France
name.surname@mines-ales.fr
[2] LIRMM CNRS 5506, 161 Rue Ada, 34392 Montpellier, France
name.surname@lirmm.fr

Abstract. The number of documents is growing exponentially with the rapid expansion of the Web. The new challenge for Internet users is now to rapidly find appropriate data to their requests. Thus information retrieval, automatic classification and detection of opinions appear as major issues in our information society. Many efficient tools have already been proposed to Internet users to ease their search over the web and support them in their choices. Nowadays, users would like genuine decision tools that would efficiently support them when focusing on relevant information according to specific criteria in their area of interest. In this paper, we propose a new approach for automatic characterization of such criteria. We bring out that this approach is able to automatically build a relevant lexicon for each criterion. We then show how this lexicon can be useful for documents classification or segmentation tasks. Experiments have been carried out with real datasets and show the efficiency of our proposal.

Keywords: Criteria characterization, Mutual Information, Classification, Segmentation.

1 Introduction

With the development of web technologies, always increasing amounts of documents are available. Efficient tools are designed to help extracting relevant information. Information and Communication Technologies are thus a kernel factor in developing our modes of organisation, if not our societies. Everybody has already visited recommendation sites to consult opinions of other people before choosing a movie or a e-business website. Our goal in this paper is to automatically identify all parts in a document that are related to a same center of interest, i.e. a specific criterion in the area of interest of an Internet user. In this paper, we present a new automatic approach named *Synopsis* which tags items of texts according to predefined criteria. First, *Synopsis* builds a lexicon containing words that are characteristic of a criterion and words that are not characteristic of this criterion from a set of documents merely downloaded using

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 457–465, 2011.

a web search engine (google for example). The way this lexicon is built is of great influence in documents classification and segmentation activities.

The paper is organized as follows. Section 2 first describes the main principles of Synopsis approach. Then, a detailed description of Synopsis is provided step by step. Section 3 presents the different experiments we have carried out. A state of the art is presented in section 4 to facilitate the understanding of results and section 5 finally presents some concluding remarks and future work.

2 The Synopsis Approach

In this section, an overview of *Synopsis* process is first presented. The detail of each step (document retrieving, words extraction, ...) required by the approach is then provided.

2.1 General Presentation

All along this paper, "movie" is our application domain and we focus on two criteria: *actor* and *scenario*. The general architecture of the approach is described in figure 1.

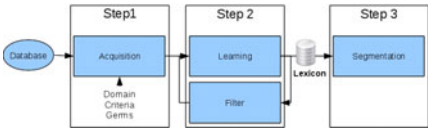


Fig. 1. General architecture

• Step 1 consists in defining the domain and the criteria. For each criterion, the user just needs to give a set of words called *germs* to specify what this criterion intends to be. For example, the germs for our two criteria may be :

scenario → *adaptation, narrative, original screenplay, scriptwriter, story, synopsis*
actor → *acting , actor, casting, character, interpretation, role, star*

A corpus of documents is built for each germ of a specific criterion (relatively to a domain) by querying a search engine for documents containing at least one occurrence of this germ. The final goal is to identify words which are correlated to one germ. At the end of this step, the corpus of documents named *class* of the criterion is the union of all the corpora related to criterion's germs. Similarly, a second corpus is built for this criterion: this time, the search engine gathers documents containing none of the criterion's germs. Intuitively, this corpus named *anti-class* intends to give a better characterization of the criterion.

• Step 2 intends to identify the word representative (resp. non- representative) of the criterion from the lemmatized texts from Step 1. This is achieved by studying the frequency of words strongly correlated with germs of the criterion. The method especially focuses on words which are close to the seeds. The

following assumption is thus made: the more frequently a word appears near a germ, the more likely it is to characterize the criterion. These words are identified by using a text window centered on germs (*c.f.* Section 2.2). The size of the window is set to an a priori given number of common nouns (grammatically speaking). Processing documents in both corpora (class and anti-class) provides a set of words close to germs with their frequency in the class or words with their frequency in the anti-class. Four kinds of words can then be defined:

1. Very frequent words in *class* and poorly frequent in *anti-class*;
2. Very frequent words in *anti-class* and poorly frequent in *class*;
3. Very frequent words in both corpora ;
4. Poorly frequent words in both corpora.

In the first two cases, information from frequencies is sufficient to take a decision upon the word's membership of the criterion. A score that characterizes the degree of membership of the criterion is computed from the word's frequency. Case 3 illustrates words which are not discriminating because they belong to both classes and are therefore to be eliminated. In the last case, the corpora of documents related to the word cannot be used to decide whether the word belongs or not to the criterion. In that latter case, another additional stage shall be performed to get new documents related to poorly frequent words. However, because of the large number of words generally obtained at this stage, a first filtering phase is performed. This one is made by applying a web measure named *Acrodef* [8]. The remaining words, named *candidate word*, are then processed one by one (*c.f.* Section 2.3).

• Step 3 consists in using the lexicon provided in Step 2 for classification, indexation or segmentation relatively to the criterion.

2.2 Characterization Criteria

Acquisition. As explained in section 2.1, the first step consists in acquiring and pre-processing Web documents. Acquisition is done by using a search engine. For each germ g of a criterion C the system retrieves about 300 different documents containing words: germ g and domain D (e.g. actor and movie). The resulting set of documents for all germs of criterion C defines the criterion's *class*. Similarly, the system seeks about 300 documents containing none of the germs of the criteria C . This set of documents defines the anti-class of criterion C. It is designed to limit the expansion of the class. Thus, the class of a criterion C is composed of about $n * 300$ documents (where n is the number of germs) and its anti-class is of about 300 documents. All the documents in the class and anti-class are then processed using a morpho-syntactic analyzer for lemmatization purposes.

Learning step. Our learning process is based on the following assumption: Words that characterize a criterion are often associated with this criterion. Thus, we are looking for words that are strongly correlated with germs, i.e. words that

frequently appear very close to germs. To identify those words we define windows centered on germs in each document t. A window is formally defined as follows:

$$F(g, sz, t) = \{m \in t / d^t_{NC}(g, m) \leq sz\} \tag{1}$$

where g is the germ, sz represents the given size of the window, and $d^t_{NC}(g, m)$ is the distance between m and g: It counts the number of grammatical *common nouns* words between m and g. We focus on the common nouns as they are known to be meaningful words in texts [5].

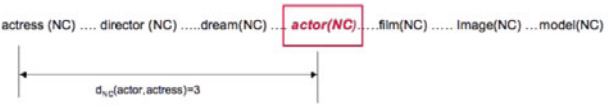


Fig. 2. An example for a window of size 3

Example 1. *Figure 2 shows a sample window of size 3 centered on the germ* actor*: there are 3 common names on its left (*actress, director, dream*) and 3 common names on its right (*film, image, model*).*

When a word m appears in a window $F(g, sz, t)$, we have then to take into account its relative position (explanation below) with regard to germ g at the center of the window. This is done by introducing the notion of influence: $I(m, g, sz, t)$ (for a window size sz in text t) :

$$I(m, g, sz, t) = \begin{bmatrix} 0 \textbf{ if } m \notin F(g, sz, t) \\ h(d^t_*(m, g)) \textbf{ if } m \in F(g, sz, t) \end{bmatrix} \tag{2}$$

where $d^t_*(m, g)$ is the distance between words m and g regardless of the grammatical nature of words. The h function is used to balance the weight associated to a word according to the distance that separates it from the germ. We also consider a notion of semi-Gaussian to smooth the effect of words nearby the edges of the window. This is done by normalizing the size of the window for distance $d^t_*(m, g)$ in order to get an interval centered on g with radius 1. Let us note l and r respectively the words that are the left and right edges of the window. Thus, h is defined by:

$$h = \begin{cases} gauss\left(\frac{d^t_*(g,M)}{d^t_*(g,l)}, \mu, \sigma\right) & \text{for a word to the left of } g. \\ gauss\left(\frac{d^t_*(g,M)}{d^t_*(g,r)}, \mu, \sigma\right) & \text{for a word to the right of } g. \end{cases}$$

$$gauss(x, \mu, \sigma) = \exp - \frac{(x - \mu)^2}{2\sigma^2} \tag{3}$$

Representativeness. For each word M previously established, we will now compute its representativeness which is a couple of values $(X, \overline{X})$. Where X is the representativeness component regarding to the *class* and $\overline{X}$ is representativeness

component relatively to the *anti-class*. Let $\mathcal{O}(M,T)$ be the set of occurrences of the word M in a text T. Let S be the set of germs for the studied criterion. Then the components of the representativeness are computed as follows:

$$X(M,sz) = \sum_{g \in \mathcal{S}} \sum_{t \in \mathcal{T}(g)} \sum_{\gamma \in \mathcal{O}(g,t)} \sum_{m \in \mathcal{O}(M,t)} I(m,g,sz,t) \tag{4}$$

$X(M,sz)$ is thus the cumulative impact of all germs g of the criterion on word M in all the texts of the *class*.

$$\overline{X}(M,sz) = \sum_{t \in anti-class} \sum_{m \in \mathcal{O}(M,t)} I(m,D,sz,t) \tag{5}$$

$\overline{X}(M,sz)$ is thus the cumulative impact of germ of the domain D on the word M in all documents of the *anti-class*. As the size of the *anti-class* is quite different from the one of the *class*, the values X and $\overline{X}$ are normalized according to the number of germs in the criterion and to the number of documents in both corpora. Both components of representativeness of a word are isomorphic to a frequency respectively in the *class* for X and in the *anti-class* for $\overline{X}$. They are used as such in the following.

Validation of candidate words. To determine whether a candidate word helps or not in criteria discrimination, we apply a web mesure like $AcroDef_{IM3}$ [8] whose objective is to evaluate the correlation between two words in a given context. All words retained by the $AcroDef_{IM3}$ filtering step are then processed to get a new value of X and $\overline{X}$ for each of them. To do that, for each candidate word, a set of documents is downloaded from the Web (*c.f.* Section 2.3). Each of them must contain both the candidate word and one of the germs of the criteria.

Discrimination. By using the value of representativeness X and $\overline{X}$ computed as previously explained, we can now define a score for each word as follows:

$$Sc(M,sz) = \frac{(X(M,sz) - \overline{X}(M,sz))^3}{(X(M,sz) + \overline{X}(M,sz))^2} \tag{6}$$

Table 1. Example of common lexicon for criterion *actor*

Term	film	actress	theater	Sam Wothington
Score	1040	450	-460	2640

The scored obtained for each words M are stored in the lexicon associated to the criterion. This is illustrated in Figure 1 for criterion *actor* in domain *movie* after processing the *candidate words* (c.f. Section 2.2). We can see that the score of *Sam Wothington* is now very high while *theater* is low. Thus *Sam Wothington* is representative of the criterion *actor* in the domain *movie* whereas *theater* is no more considered since stands for a general term. These results correspond to the expected behavior of our assessment system. The lexicon may now be used in Step 3 for classification, segmentation, or indexation.

2.3 Resolve Candidate Words

As candidate words are both infrequent in the domain and in the criterion it is much harder to obtain representative samples of texts including this word. Then, X and $\overline{X}$ for a candidate word are irrelevant. We have developed a method to collect texts containing the candidate word and we have introduced a web measure to finally get X and $\overline{X}$ values for the candidate word that are normalized relatively to values of the other words in the lexicon.

3 Experiments

In order to analyze *Synopsis* performances, several experiments have been carried out with *Google* as search engine and *TreeTagger* [9] as lemma and morpho-syntaxic parser. The field of experiment is the one described all along this paper: *Movie* is the domain, actor and scenario are the criteria. Classification and Segmentation tests were performed using a test corpus containing about 700 sentences that have been labeled by experts for each criterion. This corpus mainly contains film criticisms originated from different sources: blogs, journalism reviews...

3.1 System Performance

The following experiments are performed in a classification context. They highlight interest of enriching the lexicon with candidate words. The *test corpus* is the one described above. Validation tests are based upon conventional indicators: recall, precision, and F-measure. They are defined as follows:

Table 2. System performance (CW^* : *Candidate Word Resolution*)

***Synopsis* Performance**				
	***actor* criteria**		***scenario* criteria**	
	Without CW^*	With CW^*	Without CW^*	With CW^*
F-measure	0.39	**0.89**	0.49	**0.77**
precision	0.25	**0.87**	0.59	**0.67**
recall	0.94	**0.90**	0.41	**0.88**

Table 2 highlights interest of enriching the lexicon with candidate words. Note that if Learning is only based on frequent word ("Without candidate word" column) lots of details are lost. Thus the F-measure gets very low value (*i.e.* 0.39 for *actor* and 0.49 for *scenario*). As soon as candidate words are taken into account ("With candidate word" column) the F-measure rapidly increases (0.89 for *actor* and 0.77 for *scenario*).

3.2 Comparison with Two Standard Tools in the Context of Segmentation: *C99* and *TextTiling*

This section is devoted to the comparison of our approach *Synopsis* with other ones usually used for segmentation tasks: *C99* and *TextTiling*. As these approaches do not associate labels with the segment they identify, we are compelled to associate labels to the segments to make our comparison. The affectation of labels (*actor* or *non actor* in this experiment) to each segment is carried out in such a way that *C99* and *TextTiling* obtain the maximal score for any of the three assessment rates: *precision*, *recall* or *F-measure*. These maximal values are then compared to *Synopsis* results in Table 3.

Table 3. Comparison of the three segmenters for criteria *actor*

	for actor			**for scenario**		
Maximize	C99	TextTiling	Synopsis	C99	TextTiling	Synopsis
F-measure	0.39	0.43	**0.89**	0.36	0.38	**0.77**
Precision	0.25	0.29	**0.87**	0.25	0.28	**0.67**
Recall	0.80	0.81	**0.90**	0.65	0.65	**0.88**

Synopsis clearly provides much better performances. F-measure and accuracy are always higher than those that could ever be obtained with *C99* or *TextTiling*. However this result is to be moderated since both usual segmenters do not use any initial Learning corpus.

4 Related Work

The approach described in this paper is able to identify fragments of texts in relation with a given topic of interest or domain. Thus, our work may appear rather close to segmentation of texts and thematic extraction approaches. Segmentation is the task that identifies topic changes in a text. A segment or extract is then supposed to have a strong internal semantic while being without semantic links with its adjacent extracts. *Synopsis* not only detects topic changes in the text but also builds the subset of extracts that are related to a same topic, i.e the text is not only segmented but also indexed by the criteria defined over the domain. It results in the identification of the thematic structure of the text [6]. As many studies, our approach relies on statistical methods. For example, *TextTiling* studies the distribution of terms according to criteria [4]. Analyzing the distribution of words is a widely spread technique in segmentation process [7]. Other methods, such as *C99* approach, are based on the computation of similarities between sentences in order to detect thematic ruptures [1]. Note that segmentation approaches have, in our point of view, a major weakness: They cannot identify the topic an extract deal with. As a consequence, segmentation approaches cannot identify topic repetition in a document. Techniques issued from text summarization may in turn identify parts of a document that are related with the dominant topic of the document [2]. Identifying segments of text

related to a set of criteria in a given domain, i.e. identifying the thematic structure of a document with *Synopsis* is yet another challenge. Most of automatic summarization techniques are based upon supervised training methods and thus require a high degree of human intervention to create a training corpus. Our unsupervised framework is a major asset of *Synopsis*. The only required intervention of human beings consists in providing a small set of germ words for each criterion of interest in the domain.

5 Conclusion

In this paper, we have proposed a new approach *Synopsis* to identify the thematic structure of a text. It consists in identifying segments of texts related to criteria defined upon a domain. *Synopsis* is to be considered as a unsupervised approach since the only human intervention consists in providing a subset of germ words for each criterion. We have discussed the interest of the anti-class concept. We have demonstrated that partitioning words into words related to a criterion and words absent from this criterion provide safer classification results. In order to eliminate as quickly as possible noisy words, we have shown that mutual influence measures like *Acrodef* could help in order to minimize the number of words that require a more detailed analysis. Finally, experiments have highlighted that *Synopsis* performances are relevant both in classification and segmentation. Prospects related to this work are numerous. First, we want to extend the approach in order that *Synopsis* could incrementally learn new words. This step is a major challenge when studying a given domain. Indeed, let us consider the case of proper nouns. As earlier discussed, classification is significantly improved when proper nouns are included into the lexicon. Secondly, we wish to extend our approach by extracting opinions expressed in excerpts of specific criteria (that is the reason why subtopic of a domain are named criteria in *Synopsis*). In previous work [3], we have demonstrated that the way opinions are expressed depend on the domain: opinions detection thus appears as an obvious further development of *Synopsis*.

References

1. Choi, F.Y.Y.: Advances in domain independent linear text segmentation. In: Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics, vol. 23, pp. 26–33 (2000)
2. Chuang, W.T., Yang, J.: Extracting sentence segments for text summarization: A machine learning approach. In: Proceedings of the 23rd ACM SIGIR, pp. 152–159 (2000)
3. Harb, A., Plantie, M., Dray, G., Roche, M., Trousset, F., Poncelet, P.: Web opinion mining: how to extract opinions from blogs? In: International Conference on Soft Computing as Transdisciplinary Science and Technology (2008)
4. Hearst, M.A.: Texttiling: segmenting text into multi-paragraph subtopic passages. ACM 23, 33–64 (1997)

5. Kleiber, G.: Noms propres et noms communs: un problème de dénomination. Meta, 567–589 (1996)
6. McDonald, D., Hsinchun, C.: Using sentence-selection heuristics to rank text segments in txtractor. In: JCDL 2002, pp. 28–35 (2002)
7. Reynar, J.C.: Topic segmentation: Algorithms and applications. PhD thesis (2000)
8. Roche, M., Prince, V.: AcroDef: A quality measure for discriminating expansions of ambiguous acronyms. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 411–424. Springer, Heidelberg (2007)
9. Schmid, H.: Treetagger. In TC project at the institute for Computational Linguistics of the University of Stuttgart (1994)

A Theoretical and Experimental Comparison of Algorithms for the Containment of Conjunctive Queries with Negation

Khalil Ben Mohamed, Michel Leclère, and Marie-Laure Mugnier

University of Montpellier II, France,
{benmohamed,leclere,mugnier}@lirmm.fr

Abstract. We tackle the containment problem for conjunctive queries with negation, which takes two queries q_1 and q_2 as input and asks if q_1 is contained in q_2. A general approach for solving this problem consists of considering all completions of q_1 (intuitively these completions represent all canonical databases that satisfy q_1) and checking if each completion yields the same answer on q_2. Since the total number of completions of q_1 is exponential in the size of q_1, several proposals have aimed at reducing the number (and the size) of the completions checked. In this paper, we propose a unifying framework for comparing algorithms following this general approach and define two kinds of heuristics for exploring the space of completions. Combining these heuristics with both classical kinds of traversals, i.e., depth-first and breadth-first, we obtain four algorithms that we compare experimentally with respect to running time on difficult instances of the containment problem.

1 Introduction

This paper is devoted to the containment problem for conjunctive queries with negation (denoted CQC$^\neg$ hereafter). Stated generally, the query containment problem takes two queries q_1 and q_2 as input, and asks if q_1 is contained in q_2 (noted $q_1 \sqsubseteq q_2$), i.e., if the set of answers to q_1 is included in the set of answers to q_2 for all databases (e.g. [AHV95]). It has long been recognized as a fundamental database problem, which underlies many tasks such as query evaluation and optimization [CM77] [ASU79], rewriting queries using views [Hal01] or detecting independence of queries from database updates [LS93]. Positive conjunctive queries are considered as the basic database queries [CM77]. Conjunctive queries with negation extend this class with negation on subgoals.

When only positive conjunctive queries are considered, query containment checking is NP-complete [AHV95]. It can be solved by checking if there is a homomorphism from q_2 to q_1. When atomic negation is considered, the problem becomes much more complex: it is π_2^P-complete[1] [FNTU07] [CM09]. A general approach for solving it, first presented in [LS93] for conjunctive queries with inequalities and adapted in [Ull97] for conjunctive queries with negation, consists in considering all ways of completing q_1 with positive information. Intuitively, this amounts to generating representatives of all

[1] $\pi_2^P = co\text{-}(NP^{NP})$.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 466–480, 2011.

database instances that satisfy q_1. In [LM07], as in the present paper, negative information is also explicitly added. Such queries obtained from q_1 by adding missing information, either positively or negatively, are called completions of q_1 (and total completions if no more information can be added) hereafter. Then $q_1 \sqsubseteq q_2$ if and only if there is a homomorphism from q_2 to each total completion of q_1. However, the number of (total) completions of q_1 is exponential in the size of q_1. Several proposals have aimed at reducing the number and the size of completions ([WL03],[LM07],[BLM10a]). These proposals had not been compared yet, neither in depth from a theoretical side nor experimentally.

In this paper, we first define a unifying abstract framework, which allows us to express existing algorithms, hence to explain the principles underlying these algorithms, and to propose new algorithms. This framework relies on a space of completions of q_1 and the CQC$^\neg$ problem is reformulated as a search problem in this space. The family of algorithms we consider here perform a traversal of this space, thus build a search tree. But they essentially differ in the definition of the tree (i.e., how the children of a node are defined) and the strategy for generating the tree (i.e., in a depth-first or breadth-first way). Our second contribution consists in the experimental comparison of these algorithms. More precisely, our analysis yields four algorithm schemes, with both being close to existing proposals ([WL03],[LM07]); we implemented them, by making all algorithms benefiting from further specific heuristics experimentally evaluated in one of our former papers [BLM10a]. The experiments were made on random problem instances (i.e., pairs of conjunctive queries with negation) known to be difficult.

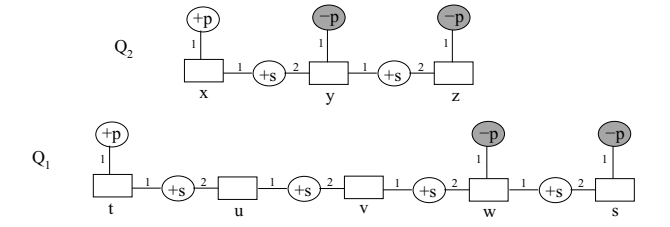
Paper layout. Section 2 is devoted to basic notions. Section 3 defines the abstract framework and containment checking methods within this framework. Section 4 presents algorithms and compare them experimentally. In section 5 the relationships of existing proposals with the evaluated algorithms are specified. Section 6 concludes this work and outlines perspectives.

2 Preliminaries

A *conjunctive query with negation ($CQ^\neg$)* is of the form: $q = ans(x_1, \ldots, x_q) \leftarrow p_1, \ldots, p_n, n_1, \ldots, n_m$, where each p_i (resp. n_i) is a positive (resp. negative) *subgoal*, $1 \leq n + m$, and *ans* is a special relation (which defines the answer part of the query). The left part of the query is called its *head* and the right part is its *body*. Each subgoal is of form $r(t_1, \ldots, t_k)$ (positive subgoal) or $\neg r(t_1, \ldots, t_k)$ (negative subgoal) where r is a relation and $t_1, \ldots, t_k$ is a tuple of terms (i.e., variables or constants). All variables $x_1, \ldots, x_q$ occur at least once in the body of the query. Without loss of generality, we assume that the same subgoal does not appear twice in the body of the query. A $CQ^\neg$ is *positive* if it has no negative subgoal ($m = 0$). A $CQ^\neg$ is *Boolean* if it has no variable in its head (we note *ans()*). In the following, we will focus on Boolean queries because having a non-empty *ans* part can only make the query containment problem easier.

As in [LM07], we will see $CQ^\neg$ as labeled graphs. More precisely, a $CQ^\neg$ q is represented as a bipartite, undirected and labeled graph Q, called *polarized graph (PG)*, with two kinds of nodes: term nodes and relation nodes. Each term of the query becomes a

term node, that is unlabeled if it is a variable, otherwise it is labeled by the constant itself. A positive (resp. negative) subgoal with relation r becomes a relation node labeled $+r$ (resp. $-r$) and it is linked to the nodes assigned to its terms. The labels on edges correspond to the position of each term in the subgoal (see Figure 1 for an example). For simplicity, the subgraph corresponding to a subgoal, i.e., induced by a relation node and its neighbors, is also called a *subgoal*. We note it $+r(t_1, \ldots, t_k)$ (resp. $-r(t_1, \ldots, t_k)$) if the relation node has label $+r$ (resp. $-r$) and list of neighbors $t_1, \ldots, t_k$. We note $\sim r(t_1, \ldots, t_k)$ a subgoal that can be positive or negative, i.e., $\sim \in \{+, -\}$. Subgoals $+r(t_1, \ldots, t_k)$ and $-r(u_1, \ldots, u_n)$ with the same relation but different signs are said to be *opposite*. Given a relation node label (resp. subgoal) l, $\overline{l}$ denotes the *complementary* relation node label (resp. subgoal) of l, i.e., it is obtained from l by reversing its sign. Queries are denoted by small letters (q_1 and q_2) and the associated graphs by the corresponding capital letters (Q_1 and Q_2). We note $Q_1 \sqsubseteq Q_2$ *iff* $q_1 \sqsubseteq q_2$. A PG is *consistent* if it does not contain two complementary subgoals.



$$q_1 = ans() \leftarrow p(t), s(t,u), s(u,v), s(v,w), \neg p(w), s(w,s), \neg p(s)$$
$$q_2 = ans() \leftarrow p(x), s(x,y), \neg p(y), s(y,z), \neg p(z)$$

Fig. 1. Polarized graphs associated with q_1 and q_2

A *homomorphism* h from a PG Q_2 to a PG Q_1 is a mapping from term nodes of Q_2 to term nodes of Q_1 that satisfies: (1) if a term node is labeled by a constant, then its image has the same label (otherwise there is no constraint on the label of its image); (2) if $\sim r(t_1, \ldots, t_k)$ is a subgoal in Q_2 then $\sim r(h(t_1), \ldots, h(t_k))$ is a subgoal in Q_1. This notion can be seen as an extension to negative subgoals of the well-known query homomorphism (classically defined on positive queries). When there is a homomorphism h from Q_2 to Q_1, we say that Q_2 *maps* to Q_1 by h.

If Q_2 and Q_1 have only positive subgoals, $Q_1 \sqsubseteq Q_2$ *iff* Q_2 maps to Q_1. When they contain negative subgoals, only one side of this property remains true: if Q_2 maps to Q_1 then $Q_1 \sqsubseteq Q_2$; the converse is false, as shown in Example 1.

Example 1. See Figure 1: Q_2 does not map to Q_1 but $Q_1 \sqsubseteq Q_2$. Indeed, if we "complete" q_1 with all possible cases w.r.t. the relation p, we obtain the union of four queries
$q_{1,1} = ans() \leftarrow p(t), s(t,u), s(u,v), s(v,w), \neg p(w), s(w,s), \neg p(s), p(u), p(v)$,
$q_{1,2} = ans() \leftarrow p(t), s(t,u), s(u,v), s(v,w), \neg p(w), s(w,s), \neg p(s), \neg p(u), p(v)$,
$q_{1,3} = ans() \leftarrow p(t), s(t,u), s(u,v), s(v,w), \neg p(w), s(w,s), \neg p(s), p(u), \neg p(v)$ and
$q_{1,4} = ans() \leftarrow p(t), s(t,u), s(u,v), s(v,w), \neg p(w), s(w,s), \neg p(s), \neg p(u), \neg p(v)$.
Each of the queries corresponds to a possible way of completing q_1 w.r.t. p. Q_2 maps to each of the graphs associated with them. Thus q_1 is contained in q_2.

One way to solve CQC$^\neg$ is therefore to generate all "complete" PGs obtained from Q_1 using relations appearing in Q_1, and then to test if Q_2 maps to each of these graphs.

Definition 1 (Complete graph and completion). *Let Q be a consistent PG. It is* complete*, denoted Q^c, w.r.t. a set of relations $\mathcal{P}$, if for each $p \in \mathcal{P}$ with arity k, for each k-tuple of term nodes (not necessarily distinct) $t_1, \ldots, t_k$ in Q, it contains $+p(t_1, \ldots, t_k)$ or $-p(t_1, \ldots, t_k)$. A* completion *Q' of Q is a PG obtained from Q by repeatedly adding new relation nodes (on term nodes present in Q), without yielding inconsistency. Each addition is a* completion step*. A completion of Q is called* total *if it is a complete graph w.r.t. the set of relations considered, otherwise it is called* partial*.*

Theorem 1. *[LM07] Let Q_1 and Q_2 be two PGs (Q_1 consistent), $Q_1 \sqsubseteq Q_2$* iff *Q_2 maps to all total completions of Q_1 w.r.t. the set of relations appearing in Q_1.*

3 Methods for Testing Containment

The complexity of a brute-force algorithm that would generate and test all completions of q_1 is prohibitive: $\mathcal{O}(2^{(n_{Q_1})^k \times |\mathcal{P}|} \times hom(Q_2, Q_1^c))$, where n_{Q_1} is the number of term nodes in Q_1, k is the maximum arity of a relation, $\mathcal{P}$ is the considered set of relations and $hom(Q_2, Q_1^c)$ is the complexity of checking the existence of a homomorphism[2] from Q_2 to Q_1^c.

Different tracks have been explored to reduce the number of homomorphisms performed. A first one is to reduce the number of considered total completions. [LM07] introduced the notion of *completion vocabulary* (denoted $\mathcal{V}$ in the following) which restricts the set of relations to consider for total completions: only relations appearing in opposite subgoals both in Q_2 and in Q_1 are to be considered. E.g. in Example 1 (Figure 1), $\mathcal{V} = \{p\}$.

A second studied track is to exploit partial completions. This idea, introduced in [WL03] and further developed in [LM07], aims at concluding about the containment before generating all total completions, by using some sufficient conditions on partial completions for success or failure of the containment.

3.1 The Completion Space

The completion space of a PG Q_1 (w.r.t. a given completion vocabulary) is the set of partial and total completions of Q_1 partially ordered by the relation "subgraph of". If Q_j is a descendant of Q_i, i.e., Q_i is a subgraph of Q_j, we say that Q_i *covers* Q_j. The *successors* of a completion are its immediate descendants (note that they only differ from it by one added subgoal).

Figure 2 shows the completion space of Q_1 from Example 1. Q_1 has four successors, namely the partial completions $Q_{1,1}$, $Q_{1,2}$, $Q_{1,3}$ and $Q_{1,4}$, obtained by adding respectively $+p(u)$, $+p(v)$, $-p(u)$ and $-p(v)$. From $Q_{1,1}$ we obtain two total completions $Q_{1,5}$ and $Q_{1,6}$ by adding respectively $+p(v)$ and $-p(v)$ (note that we cannot add $-p(u)$ because $+p(u)$ is present in $Q_{1,1}$), and similarly with the others $Q_{1,i}$. Finally, there are four total completions of Q_1.

[2] A brute-force algorithm for homomorphism check it in $\mathcal{O}(n_{Q_1}^{n_{Q_2}})$, where n_{Q_2} is the number of term nodes in Q_2.

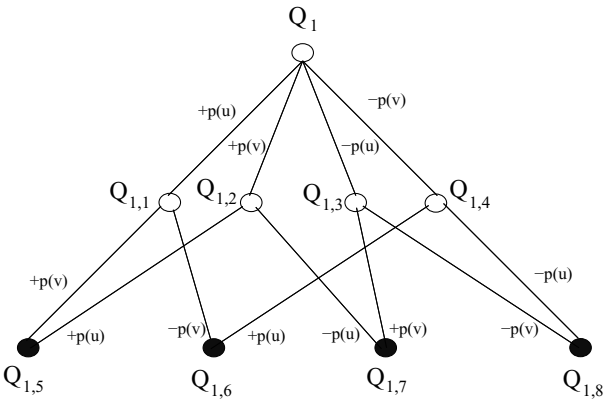


Fig. 2. The completion space of Example 1

The following notion of a covering set is fundamental in this paper:

Definition 2 (Covering set). *Let Q_1 be a (consistent) query. A* covering set *of Q_1, noted $CS(Q_1) = \{Q_{1,1}, \ldots, Q_{1,n}\}$, is a set of (partial or total) completions of Q_1 such that every total completion Q_1^c of Q_1 is covered by a $Q_{1,i}$.*

Trivial examples of $CS(Q_1)$ are $\{Q_1\}$ and the set of all total completions of Q_1. The question "does $Q_1 \sqsubseteq Q_2$ hold ?" can now be recast as "is there $CS(Q_1) = \{Q_{1,1}, \ldots, Q_{1,n}\}$ such that Q_2 maps to each $Q_{1,i}$ for $i = 1 \ldots n$?"

The methods considered in this paper can be seen as exploring the completion space of Q_1, with the aim of finding a covering set of Q_1 such that Q_2 maps to each element of this set, or deciding that there is none. More precisely, they exploit the following property:

Property 1. Let Q_1 and Q_2 be two PGs, with Q_1 consistent. For all covering set $CS(Q_1)$, it holds that: $Q_1 \sqsubseteq Q_2$ if and only if, for each $Q_{1,i} \in CS(Q_1)$, $Q_{1,i} \sqsubseteq Q_2$.

Proof. If $Q_1 \sqsubseteq Q_2$ then for each partial or total completion Q'_1 of Q_1, $Q'_1 \sqsubseteq Q_2$; this is in particular true for elements of any $CS(Q_1)$. Conversely: let Q'_1 be a total completion of Q_1. By definition of a covering set, Q'_1 is covered by at least one $Q_{1,i} \in CS(Q_1)$, thus is a total completion of $Q_{1,i}$. Since $Q_{1,i} \sqsubseteq Q_2$, there is a homomorphism from Q_2 to any total completion of $Q_{1,i}$, in particular to Q'_1. □

This simple framework yields immediate proofs for the correctness of the algorithms studied in this paper.

3.2 Sufficient Conditions for Concluding

When exploring a current completion Q'_1, two kinds of sufficient conditions for concluding about containment or non-containment can be exploited, which are both based on homomorphism checks. Note that these conditions are not symmetrical.

Sufficient condition for concluding that $Q'_1 \sqsubseteq Q_2$. A simple sufficient condition for the containment of Q'_1 in Q_2 is the existence of a homomorphism from Q_2 to Q'_1 (cf. [LM07]). When this test is successful, it allows to "prune" the descendants of Q'_1: then there is necessarily a homomorphism from Q_2 to all graphs covered by Q'_1. Note however that it does not allow to conclude that Q_1 is contained in Q_2.

Sufficient conditions for concluding that $Q_1 \not\sqsubseteq Q_2$. Failure tests try to discover that there is at least one total completion Q_1^c covered by Q'_1 that does not admit any homomorphism from Q_2. These tests lead to a global negative answer, i.e., a negative answer about the initial containment problem. These failure tests exploit the property of some special subgraphs of Q_2, that must map by homomorphism to any completion of Q_1 (including Q_1), otherwise there exists a total completion Q_1^c to which Q_2 does not map. In the following, we call them "necessary subgraphs".

A first example of necessary graphs is given in [WL03]: the subgraph of Q_2 composed of all positive subgoals of Q_2. In [LM07], a more general characterization of such graphs is given: subgraphs without "exchangeable subgoals"; checking whether a graph is without exchangeable subgoals is NP-complete [MST09], but polynomially recognizable kinds of such graphs can be used, such as *pure subgraphs* (or independent subgraphs, which moreover exploit constraints induced by constants).

Definition 3 (pure subgraph). *A PG is said to be* pure *if it does not contain opposite subgoals (i.e., each relation appears only in one form, positive or negative). A* pure subgraph *of Q_2 is a subgraph of Q_2 that contains all term nodes in Q_2 (but not necessarily all relation nodes)*[3] *and is pure.*

See Figure 1: there are two pure subgraphs maximal for the inclusion: Q_2^+ contains $+p(x)$, $+s(x,y)$ and $+s(y,z)$; Q_2^- contains $-p(y)$, $-p(z)$, $+s(x,y)$ and $+s(y,z)$.

Moreover, the notion of necessary subgraphs goes with a more constrained homomorphism test, called *compatible homomorphism*. Intuitively, a homomorphism from a necessary subgraph of Q_2 to Q_1 is "compatible" if it can be extended to a homomorphism from Q_2 to a total completion of Q_1.

Definition 4 (Compatible homomorphism). *Let Q_2 and Q_1 be two PGs and Q'_2 be a necessary subgraph of Q_2. A homomorphism h from Q'_2 to Q_1 is said to be* compatible *w.r.t. Q_2 if, for each subgoal $\sim r(t_1,\ldots,t_k)$ in $Q_2 \setminus Q'_2$, the opposite subgoal $\overline{\sim r}(h(t_1),\ldots,h(t_k))$ is* not *in Q_1, and for each pair of opposite subgoals in $Q_2 \setminus Q'_2$, respectively on $(c_1,\ldots,c_k)$ and $(d_1,\ldots,d_k)$, $(h(c_1),\ldots,h(c_k)) \neq (h(d_1),\ldots,h(d_k))$.*

Property 2. [LM07] Let Q_1 and Q_2 be two PGs and Q'_2 be a necessary subgraph of Q_2. If there is no compatible homomorphism from Q'_2 to Q_1, then $Q_1 \not\sqsubseteq Q_2$.

3.3 Exploration Heuristics of the Completion Space

The exploration of the completion space can be seen as an iterative procedure maintaining a covering set $CS(Q_1)$ and trying to find a "good" covering set, i.e., such that Q_2

[3] Note that this subgraph does not necessarily correspond to a set of subgoals because some term nodes may be isolated.

maps to each of its elements, or to show that there is none. Initially, $CS = \{Q_1\}$. At each step, the procedure performs the following:

1. pick a current completion Q'_1 in CS;
2. check if Q'_1 leads to conclude with a global failure;
3. otherwise: if Q_2 does not map to Q'_1, add to CS some successors of Q'_1, while keeping the property that CS is a covering set of Q_1.

When CS has been emptied, the global containment test succeeds. The set of built completions to which Q_2 maps can be seen as a proof that $Q_1 \sqsubseteq Q_2$.

Two ways of looking for a "good covering" set can be defined, which correspond to two exploration heuristics, called **dichotomic** and **"contradictAll"** hereafter. Note that, although not explicitly expressed as such, the proposals in [LM07] and in [WL03] can be seen as examples of these heuristics.

Before specifying them, let us consider the following notions:

Definition 5 (Missing subgoal, h-extension, h-contradiction). *Let Q_1 and Q_2 be two PGs (Q_1 consistent), Q'_2 a necessary subgraph of Q_2, and h a compatible homomorphism from Q'_2 to Q_1. Given $\sim r(t_1, \ldots, t_k)$ from $Q_2 \setminus Q'_2$, the subgoal $\sim r(h(t_1), \ldots, h(t_k))$ is said to be* missing *to Q_1 w.r.t. h if it is not in Q_1. A completion Q'_1 of Q_1 is called an* h-contradiction *if it contains the complementary of a missing subgoal w.r.t. h; otherwise it is called an* h-extension.

The dichotomic heuristic. At each step, this heuristic partitions the completion space into two (disjoint) subspaces, by generating two completions from Q'_1 (the currently considered completion of Q_1): these completions are respectively obtained by adding a subgoal and its complementary. Since completions are consistent, it follows that the sets of completions covered by these newly generated completions are disjoint.

This method can be further specified by the choice of a necessary subgraph of Q_2, a compatible homomorphism h from this subgraph to Q'_1, and a missing subgoal to Q_1 w.r.t. h, so that the newly generated completions are respectively an h-extension and an h-contradiction of Q_1.

The correctness of this method is based on the following theorem:

Theorem 2. *Let Q_1 and Q_2 be two PGs. Then $Q_1 \sqsubseteq Q_2$ iff (1) there is a compatible homomorphism from a necessary subgraph of Q_2 [e.g. a pure subgraph] to Q_1 and (2) Let h be any such homomorphism. If there is a missing subgoal to Q_1 w.r.t. h, let $\sim r(t_1, \ldots, t_k)$ be such a subgoal; then $Q'_1 \sqsubseteq Q_2$ and $Q''_1 \sqsubseteq Q_2$, where Q'_1 (resp. Q''_1) is the h-contradiction (resp. h-extension) obtained from Q_1 by adding $\overline{\sim r}(t_1, \ldots, t_k)$ (resp. $\sim r(t_1, \ldots, t_k)$).*

Proof. Follows from Properties 1 and 2, and the fact that $\{Q'_1, Q''_1\}$ is a covering set of Q_1. □

Note that, if h is directly a homomorphism from Q_2 to Q_1, then there is no missing subgoal, and condition (2) is fulfilled.

The completion space is thus explored as a binary tree with Q_1 as root.

Figure 3 illustrates this method on Example 1, with Q_2^- as the necessary subgraph. Let $h_1 = \{x \mapsto v, y \mapsto w, z \mapsto s\}$ from Q_2^- to Q_1; $Q_{1,1}$ and $Q_{1,2}$ are built from Q_1, respectively by adding $+p(v)$ and $-p(v)$. Q_2 maps to $Q_{1,1}$, thus there is no need to complete $Q_{1,1}$. Q_2 does not map to $Q_{1,2}$: let $h_2 = \{x \mapsto u, y \mapsto v, z \mapsto w\}$ from Q_2^- to $Q_{1,2}$; $Q_{1,3}$ and $Q_{1,4}$ are built from $Q_{1,2}$, respectively by adding $+p(u)$ and $-p(u)$ to $Q_{1,2}$. Q_2 maps to $Q_{1,3}$ and to $Q_{1,4}$, respectively. Finally, the set proving that Q_1 is included in Q_2 is $\{Q_{1,1}, Q_{1,3}, Q_{1,4}\}$ (and there are four total completions of Q_1 w.r.t. p).

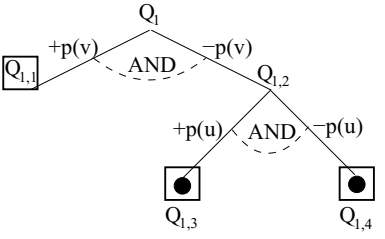


Fig. 3. A dichotomic search tree of Example 1. Each black dot represents a total completion and each square a partial one.

The "contradictAll" heuristic. This heuristic is directly related to the notion of a compatible homomorphism from a necessary subgraph of Q_2 to Q_1': at each step, it consists of choosing such a compatible homomorphism h to produce n h-contradictions, with each of them being obtained by adding to Q_1' the complementary of one of the n missing subgoals to Q_1' w.r.t. h.

The correctness of this method is based on the following theorem:

Theorem 3. *Let Q_1 and Q_2 be two PGs. Then $Q_1 \sqsubseteq Q_2$ iff (1) there is a compatible homomorphism from a necessary subgraph of Q_2 [e.g. a pure subgraph] to Q_1 and (2) Let h be any such homomorphism; then, for each missing subgoal $\sim r(t_1, \ldots, t_k)$ to Q_1 w.r.t. h, $Q_1^i \sqsubseteq Q_2$, where Q_1^i is the h-contradiction obtained from Q_1 by adding $\overline{\sim r}(t_1, \ldots, t_k)$.*

Proof. The covering of all total completions is ensured on the one hand by the construction of the n h-contradictions, and on the other hand by the h-extension $Q_1^{extension}$ (obtained from Q_1 by adding all the missing subgoals to Q_1 w.r.t. h) to which Q_2 maps. We conclude with Properties 1 and 2. □

Figure 4 illustrates this method on Example 1, with Q_2^+ as the necessary subgraph. Let $h_1 = \{x \mapsto t, y \mapsto u, z \mapsto v\}$ from Q_2^+ to Q_1; $Q_{1,1}$ and $Q_{1,2}$ are built from Q_1, respectively by adding $+p(v)$ and $+p(u)$. Note that Q_2 necessarily maps to $Q_1^{extension}$, obtained from Q_1 by adding $-p(v)$ and $-p(u)$. Q_2 maps to $Q_{1,1}$, thus there is no need to complete $Q_{1,1}$. Q_2 does not map to $Q_{1,2}$: let $h_2 = \{x \mapsto u, y \mapsto v, z \mapsto w\}$ from Q_2^+ to $Q_{1,2}$; $Q_{1,3}$ is built from $Q_{1,2}$ by adding $+p(u)$. As previously, Q_2 maps to $Q_{1_2}^{extension}$. Q_2 maps to $Q_{1,3}$. Finally, the set proving that $Q_1 \sqsubseteq Q_2$ is $\{Q_{1,1}, Q_{1,3}, Q_1^{extension}, Q_{1,2}^{extension}\}$.

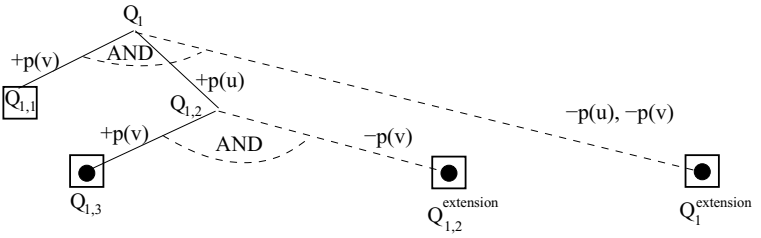


Fig. 4. A contradictAll search tree of Example 1

Nevertheless, this space exploration, which does not partition the space, raises an important problem: it might be the case that the same completion is explored several times. In the worst case, this heuristic may lead to consider more completions than the brutal method that explores all total completions. To prevent these multiple explorations, two solutions can be imagined:

1. To forbid the construction of two identical completions. Then the algorithm becomes exponential in space because explored completions have to be memorized.
2. To "merge" identical completions at the end of each completion step (this works only with a breadth-first search algorithm, see Section 4.1). But the algorithm will be locally (i.e., at each completion step) exponential in space and this merging is expensive since it requires to pairwise compare the new completions.

In our experiments, we have compared the three alternatives: no prevention of multiple explorations, Solution 1 and Solution 2.

4 Comparison of Methods and Algorithms

In this section, we present several algorithms implementing the two heuristics presented in the previous section. Both heuristics perform traversals of the completion space, which differ in the way they select the successors of a node. Moreover, there are two classical ways of performing a traversal, namely in depth-first or in breadth-first way. We will first present two generic algorithms, corresponding to these two exploration schemes. Then, by concretizing the function that selects the successors of a node, we obtain the dichotomic or contradictAll heuristics. Thus, we finally obtain four algorithms, that we compare experimentally.

4.1 Breadth-First and Depth-First Traversals

Algorithms 1 and 2 are generic algorithms that respectively perform a breadth-first and a depth-first search of the completion space. Algorithm 1 is iterative, it updates a covering set denoted by *CS*. We chose to present Algorithm 2 in a recursive way.

A negative answer to the test *if there is no homomorphism from* Q_2 allows to conclude that $Q_1 \sqsubseteq Q_2$ (cf. the sufficient condition in Section 3.2). It is an algorithmic optimization avoiding useless exploration of completions.

Subalgorithm dynamicFiltering(Q). This function corresponds to the sufficient condition for concluding that $Q_1 \not\sqsubseteq Q_2$ (cf. Section 3.2): if there is no compatible homomorphism from one (or several) necessary subgraph of Q_2 to Q then we can conclude that $Q_1 \not\sqsubseteq Q_2$.

Subalgorithm selectSuccessors(Q). This function returns a covering set of Q, which is a subset of the successors of Q. This subset depends on several variables:

1. a necessary subgraph, say Q'_2, that has to be mapped to Q;
2. a compatible homomorphism from Q'_2 to Q;
3. the chosen heuristic, i.e., dichotomic or contradictAll:
   (a) if we consider the dichotomic heuristic, we have also to choose a missing subgoal $\sim r(t_1, \ldots, t_k)$ to Q w.r.t. h. The function will then return the set $\{Q \cup \{\overline{\sim r}(t_1, \ldots, t_k)\}, Q \cup \{\sim r(t_1, \ldots, t_k)\}\}$; note that the order in which these two nodes are then explored is important, as shown in [BLM10a]: exploring first the h-contradiction (i.e., $\{Q \cup \overline{\sim r}(t_1, \ldots, t_k)\}$) is more efficient.
   (b) if we consider the contradictAll heuristic, the function will return the set $\{Q \cup \{\overline{\sim r}(t_1, \ldots, t_k)\}, \ldots, Q \cup \{\overline{\sim s}(u_1, \ldots, u_j)\}\}$ where $\sim r(t_1, \ldots, t_k), \ldots, \sim s(u_1, \ldots, u_j)$ are all the missing subgoals to Q w.r.t. h.

Algorithm 1. breadthCheck(Q_1, Q_2)

Input: two consistent PGs Q_1 and Q_2
Result: true if $Q_1 \sqsubseteq Q_2$, false otherwise
begin
  $CS \leftarrow \{Q_1\}$;
  while $CS \neq \emptyset$ **do**
    $CS' \leftarrow \emptyset$;
    foreach $Q'_1 \in CS$ **do**
      if *there is no homomorphism from* Q_2 *to* Q'_1 **then**
        if ***dynamicFiltering***(Q'_1) = *failure* **then return** *false*;
        else $CS' \leftarrow CS' \cup$ **selectSuccessors**(Q'_1);
    $CS \leftarrow CS'$;
  return *true*;
end

Finally, by combining dichotomic and contradictAll heuristics with breadth-first and depth-first searches, we obtain four algorithms: two breadth-first search ones, `dichotomicBreadthCheck` and `ContradictAllBreadthCheck`; two depth-first search ones, `dichotomicDepthCheck` and `ContradictAllDepthCheck`. In the next section we compare them experimentally.

Algorithm 2. Depthcheck(Q_1, Q_2)

```
Input: two consistent PGs Q1 and Q2
Result: true if Q1 ⊑ Q2, false otherwise
begin
    if there is no homomorphism from Q2 to Q1 then
        if dynamicFiltering(Q1) = failure then return false;
        else
            foreach Q'1 ∈ selectSuccessors(Q1) do
                if depthCheck(Q'1,Q2) = false then return false;
    return true;
end
```

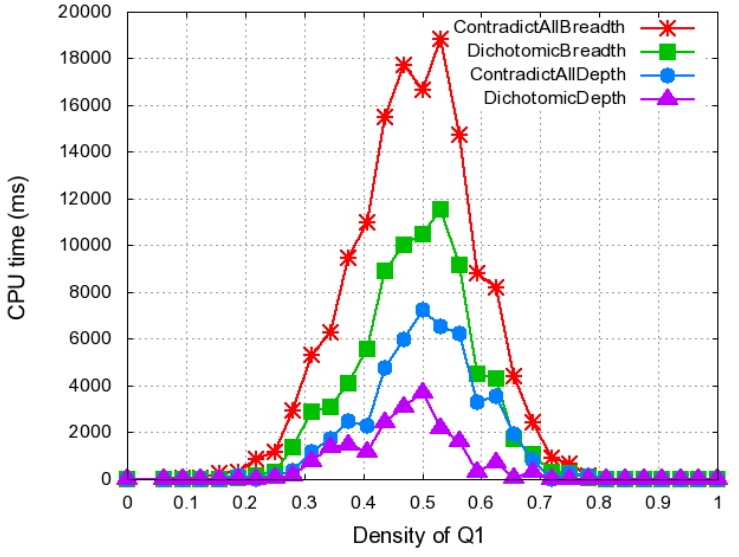
4.2 Experimental Comparison

We refer the reader to [BLM10a, BLM10b] for details about the experimental methodology. We built a random generator of polarized graphs and studied the influence of several parameters on the "difficulty" of problem instances (number of terms in the PG/query, percentage of constants, number of distinct relations, arity of these relations, density per relation, percentage of negation per relation). In the following experiments, we chose the parameter values shown to yield difficult instances. For each value of the varying parameter (density of Q_1 in the next experiments), we considered 500 instances and computed the mean search cost of the results on these instances. We also set a timeout at one minute[4].

We have made all four algorithms benefit from the improvements studied in [BLM10a]. Function **dynamicFiltering**(Q) performs filtering with all pure subgraphs of Q_2 maximal for the inclusion. Function **selectSuccessors**(Q) relies on a pure subgraph that is maximal for inclusion, say Q_2^{max}; the compatible homomorphism h from Q_2^{max} to Q is simply the first one found (as we have no criterion to choose among several such homomorphisms); in the case of dichotomic heuristic, the missing subgoal is randomly chosen among the set of missing subgoals to Q w.r.t. h. About avoiding multiple explorations, we compared experimentally the solutions proposed above and kept the best one for each algorithm: `ContradictAllBreadthCheck` uses a merging function and `ContradictAllDepthCheck` memorizes all explored completions.

Figure 5 shows the results obtained by the four algorithms on the same random CQC$^\neg$ instances. We can see that depth-first search algorithms (`dichotomicDepthCheck` and `ContradictAllDepthCheck`) are always better than breadth-first search ones. As expected, we can also see that dichotomic exploration is always better than contradictAll one (regardless of search strategy): this is due to the fact that dichotomic heuristic inherently avoids exploring twice the same completion, whereas contradictAll heuristic cannot ensure this property without a merging or memorizing function.

[4] With a timeout set at five minutes, breadth-first search algorithms lead to memory overflow.



Percentage of timeouts at a difficulty peak (with density of $Q_1 = 0.5$):
`CA-Breadth`=25%; `D-Breadth`=15%; `CA-Depth`=12%; `D-Depth`=4%.

Fig. 5. Comparison of the four algorithms

5 Relationships with Existing Algorithms

In [LM07], Leclre and Mugnier proposed a depth-first search algorithm based on the dichotomic heuristic. We optimized this algorithm in [BLM10a], that led to a refined algorithm named `recCheckPlus`. This latter algorithm is exactly `dichotomicDepthCheck`.

In [WL03], Wei and Lausen proposed a breadth-first search algorithm (denoted by *WL-algorithm* hereafter) based on the following theorem, which we reformulate in our framework:

Theorem 4. *[WL03]. Let Q_1 and Q_2 be two PGs. Then, $Q_1 \sqsubseteq Q_2$ iff (1) there is a (compatible) homomorphism from Q_2^+ to Q_1 and (2) for each such homomorphism h and for each missing subgoal $\sim r(t_1, \ldots, t_k)$ to Q_1 w.r.t. h, $Q'_1 \sqsubseteq Q_2$, where Q'_1 is the h-contradiction obtained from Q_1 by adding $\overline{\sim r}(t_1, \ldots, t_k)$.*

Theorem 3 can be seen as a generalization of Theorem 4: at point (1), it considers a compatible homomorphism from any necessary subgraph of Q_2 to Q_1 (instead of Q_2^+), and at point (2), it avoids to test all (compatible) homomorphisms at each completion step (whereas Theorem 4 proof and *WL-algorithm* explicitly use this test). More precisely, Wei and Lausen proposed a space exploration where at each step, all homomorphisms from Q_2^+ to the current completion are to be considered. The search space is then explored as a particular AND/OR tree[5], as shown on Figure 6. To prove that $Q_1 \sqsubseteq Q_2$,

[5] Strictly speaking, it is not exactly an AND/OR tree because one of the halting conditions is global (which is based on Property 2), i.e., it allows to completely stop the process.

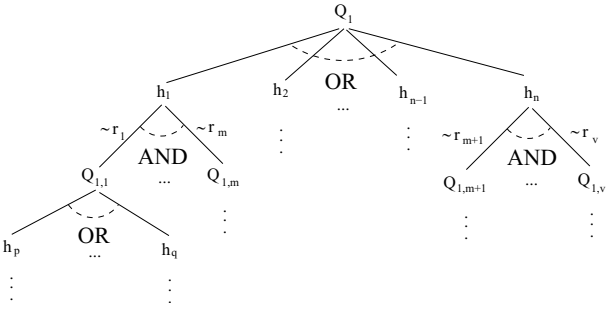


Fig. 6. "Wei and Lausen" exploration

one has to prove that $((Q_{1,1} \sqsubseteq Q_2) \wedge \ldots \wedge (Q_{1,m} \sqsubseteq Q_2)) \vee \ldots \vee ((Q_{1,m+1} \sqsubseteq Q_2) \wedge \ldots \wedge (Q_{1,v} \sqsubseteq Q_2))$.

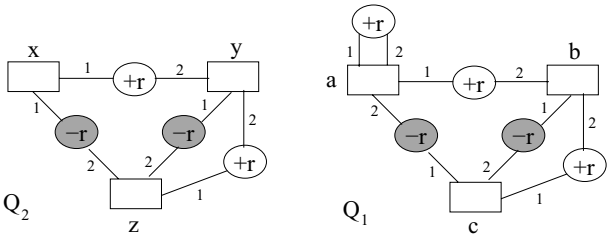
Let us comment on another aspect of *WL-algorithm*. This algorithm considers only "new" homomorphisms from Q_2^+ to Q_1' (the current completion), with the aim of avoiding multiple computation of the same homomorphisms. There are several ways of interpreting the notion of a "new" homomorphism:

1. it is new w.r.t. the path from Q_1 to Q_1', i.e., it has not been computed during the generation of this path;
2. it is new w.r.t. the subtree composed of the ancestors of Q_1' and their brothers;
3. it is new w.r.t. the entire tree.

The first possibility is necessarily fulfilled, because all explored completions are h-contradictions. Indeed, added subgoals contradict homomorphisms used throughout the path from Q_1 to Q_1'. A new homomorphism according to the second definition is such that at least one subgoal in Q_2^+ is mapped to the subgoal added at the previous completion step. More precisely, let Q_1' be a completion obtained by adding $+r(e_1, \ldots, e_k)$. A homomorphism h from Q_2^+ to Q_1' is said *new* if there is $+r(t_1, \ldots, t_k)$ in Q_2^+ such that $+r(h(t_1), \ldots, h(t_k)) \in Q_1'$ and $h(t_1), \ldots, h(t_k) = e_1, \ldots, e_k$. However, this definition of a new homomorphism is unsatisfactory for two main reasons:

1. It does not avoid multiple explorations of the same completion. In Figure 6 for example, completions $Q_{1,m}$ and $Q_{1,m+1}$ could be obtained by adding the same subgoal.
2. It makes *WL-algorithm* incomplete, i.e., this algorithm can miss solutions. Queries of Figure 7 illustrate this problem: $Q_1 \sqsubseteq Q_2$ whereas *WL-algorithm* concludes that $Q_1 \not\sqsubseteq Q_2$, because at the second completion step it does not find any **new** homomorphism (but it would continue if it considered all homomorphisms).

The last possibility makes *WL-algorithm* incomplete as well: the completion process stops whereas it should continue by "reusing" some homomorphisms of other paths (the previous counterexample works here too: at the second completion step, there are no "new" homomorphisms w.r.t. the entire tree).



$$q_1 = ans() \leftarrow r(a,a), r(a,b), r(c,b), \neg r(b,c), \neg r(c,a)$$
$$q_2 = ans() \leftarrow r(x,y), r(z,y), \neg r(x,z), \neg r(y,z)$$

Fig. 7. A counterexample to the "new homomorphism" property

In light of this analysis, it appears that homomorphism novelty is not a good notion for the needs of the algorithm. It has to consider the novelty of a completion.

In summary, both `contradictAllDepthCheck` and `contradictAllBreadthCheck` are implementations of (a generalization of) the theorem of [WL03]. Algorithm `contradictAllBreadthCheck` can be seen as a clean implementation of the algorithm proposed in the Appendix of [WL03]. Moreover, it is expressed in a very simple way, which allows to easily check its correctness. Nevertheless, algorithm `ContradictAllDepthCheck`, which is as simple to express, is better (cf. Figure 5).

Let us end this section by briefly mentioning another algorithm proposed in [BLM10a]. Its way of exploring the space is totally different: it builds a *candidate* covering set at once, and then check if this set is actually a covering set by transforming it into a propositional logical formula and checking its unsatisfiability. Then, $Q_1 \sqsubseteq Q_2$ if and only if this formula is unsatisfiable. Nevertheless, this algorithm has been experimentally shown less efficient than `dichotomicDepthCheck` on difficult instances.

6 Conclusion

In this paper, we propose a unifying framework for comparing algorithms solving CQC$^\neg$, and define two kinds of heuristics: dichotomic and contradictAll. Combining these heuristics with both classical kinds of traversals, i.e., depth-first and breadth-first, we obtain four algorithms. We compare them experimentally and show that the depth-first search algorithm with dichotomic heuristic (`dichotomicDepthCheck`) is more efficient than the three others.

Real-world queries expressed by a user generally contain constants. In our experiments, we considered queries without any constants because we focused on difficult instances. Moreover, we checked that CQC$^\neg$ difficulty decreases very quickly with the increasing of the percentage of constants. As for further work, we will study how constants can be exploited in algorithms, with the aim of drastically increasing the size of queries that can be processed within reasonable time.

Another perspective would be to compare `dichotomicDepthCheck` with logical provers solving problems of the same complexity class (Π_2^p-complete), such as the problem *2-QBF* (e.g. [GW99]).

References

[AHV95] Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases: The Logical Level. Addison-Wesley, Reading (1995)

[ASU79] Aho, A.V., Sagiv, Y., Ullman, J.D.: Equivalences among relational expressions. SIAM J. Comput. 8(2), 218–246 (1979)

[BLM10a] Mohamed, K.B., Leclére, M., Mugnier, M.-L.: Containment of conjunctive queries with negation: Algorithms and experiments. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010. LNCS, vol. 6262, pp. 330–345. Springer, Heidelberg (2010)

[BLM10b] Ben Mohamed, K., Leclère, M., Mugnier, M.-L.: Deduction in existential conjunctive first-order logic: an algorithm and experiments. Technical Report RR-10010, LIRMM (March 2010)

[CM77] Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational databases. In: 9th ACM Symposium on Theory of Computing, pp. 77–90 (1977)

[CM09] Chein, M., Mugnier, M.-L.: Graph-based Knowledge Representation and Reasoning—Computational Foundations of Conceptual Graphs. Advanced Information and Knowledge Processing. Springer, Heidelberg (2009)

[FNTU07] Farré, C., Nutt, W., Teniente, E., Urpí, T.: Containment of conjunctive queries over databases with null values. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 389–403. Springer, Heidelberg (2006)

[GW99] Gent, I.P., Walsh, T.: Beyond np: the qsat phase transition, pp. 648–653. AAAI Press, Menlo Park (1999)

[Hal01] Halevy, A.Y.: Answering queries using views: A survey. VLDB Journal 10(4), 270–294 (2001)

[LM07] Leclère, M., Mugnier, M.-L.: Some algorithmic improvements for the containment problem of conjunctive queries with negation. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 404–418. Springer, Heidelberg (2006)

[LS93] Levy, A.Y., Sagiv, Y.: Queries independent of updates. In: VLDB 1993: Proceedings of the 19th International Conference on Very Large Data Bases, pp. 171–181. Morgan Kaufmann Publishers Inc., San Francisco (1993)

[MST09] Mugnier, M.-L., Simonet, G., Thomazo, M.: On the complexity of deduction in existential conjunctive first-order logic (long version). Technical Report RR-09026, LIRMM (September 2009)

[Ull97] Ullman, J.D.: Information Integration Using Logical Views. In: Afrati, F.N., Kolaitis, P.G. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 19–40. Springer, Heidelberg (1996)

[WL03] Wei, F., Lausen, G.: Containment of conjunctive queries with safe negation. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 343–357. Springer, Heidelberg (2002)

Data Integration over NoSQL Stores Using Access Path Based Mappings

Olivier Curé[1], Robin Hecht[2], Chan Le Duc[3], and Myriam Lamolle[3]

[1] Université Paris-Est, LIGM, Marne-la-Vallée, France
ocure@univ-mlv.fr
[2] University of Bayreuth, Bayreuth, Germany
robin.hecht@uni-bayreuth.de
[3] LIASD Université Paris 8 - IUT de Montreuil
{chan.leduc,myriam.lamolle}@iut.univ-paris8.fr

Abstract. Due to the large amount of data generated by user interactions on the Web, some companies are currently innovating in the domain of data management by designing their own systems. Many of them are referred to as NoSQL databases, standing for 'Not only SQL'. With their wide adoption will emerge new needs and data integration will certainly be one of them. In this paper, we adapt a framework encountered for the integration of relational data to a broader context where both NoSQL and relational databases can be integrated. One important extension consists in the efficient answering of queries expressed over these data sources. The highly denormalized aspect of NoSQL databases results in varying performance costs for several possible query translations. Thus a data integration targeting NoSQL databases needs to generate an optimized translation for a given query. Our contributions are to propose (i) an access path based mapping solution that takes benefit of the design choices of each data source, (ii) integrate preferences to handle conflicts between sources and (iii) a query language that bridges the gap between the SQL query expressed by the user and the query language of the data sources. We also present a prototype implementation, where the target schema is represented as a set of relations and which enables the integration of two of the most popular NoSQL database models, namely document and a column family stores.

1 Introduction

In [8], several database experts argued that Relational Data Base Management Systems (RDBMS) can no longer handle all the data management issues encountered by many current applications. This is mostly due to (i) the high, and ever increasing, volume of data needed to be stored by many (web) companies, (ii) the extreme query workload required to access and analyze these data and (iii) the need for schema flexibility.

Several systems have already emerged to propose an alternative to RDBMS and many of them are categorized under the term NoSQL, standing for 'Not only SQL'. Many of these databases are based on the Distributed Hash Table (DHT)

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 481–495, 2011.

model which provides a hash table access semantics. That is, in order to access or modify an object data, a client is required to provide the key of that object and then the database will lookup the object using an equality match to the required attribute key. The first implementations where developed by companies like Google and Amazon with respectively Bigtable [1] and Dynamo [3]. These systems influenced the implementation of several open source systems such as Cassandra[1], HBase[2], etc. Nowadays, the NoSQL ecosystem is relatively rich with several categories of databases: column family (e.g. Bigtable, HBase, Cassandra), key/value (e.g. Dynamo, Riak[3]), document (e.g. MongoDB[4], CouchDB[5]) and graph oriented (e.g. InfiniteGrap[6], Neo4J[7]). Most of these systems share common characteristics by aiming to support scalability, availability, flexibility and to ensure fast access times for storage, data retrieval and analysis.

In order to meet some of these requirements, NoSQL database instances are designed to reply efficiently to the precise needs of a given application. Note that a similar approach, named denormalization [4], is frequently encountered for application using relational databases. Nevertheless, it may be required to combine the data stored in several NoSQL database instances into a single application and at the same time to leave them evolve with their own applications. This combination of data coming from different sources corresponds to the notion of a data integration system presented in [5]. Yet, several issues emerge due to the following NoSQL characteristics: (i) NoSQL categories are based on different data models and each implementation within a category may have its own specificities. (ii) There does not exist a common query language for all NoSQL databases. Moreover, most systems only support a procedural definition of queries. (iii) The NoSQL ecosystem is characterized by a set of heterogeneous data management systems, e.g. not all databases support indexing. (iv) The denormalized aspect of NoSQL databases makes query performance highly dependent on access paths.

In this paper, we present a data integration system which is based on the assumptions of Figure 1. The target schema corresponds to a standard relational model. This is motivated by the familiarity of most end-users with this data model and its possibility to be queried with the SQL language. The sources can either correspond to a set of column family, document and key/value stores as well as to standard RDBMS.

To enable the querying of NoSQL databases within a data integration framework we propose the following contributions. (1) We define a mapping language between the target and the sources which takes into account the denormalization aspect of NoSQL databases. This is materialized by storing preferred access

[1] http://cassandra.apache.org/
[2] http://hbase.apache.org/
[3] http://www.basho.com/
[4] http://www.mongodb.org/
[5] http://couchdb.apache.org/
[6] http://www.infinitegraph.com/
[7] http://neo4j.org/

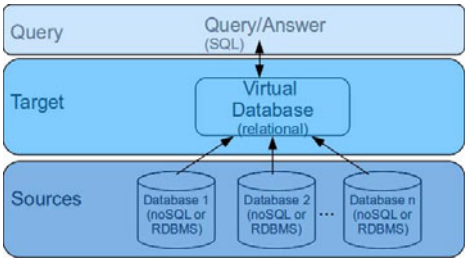


Fig. 1. Data integration overview

paths for a given mapping assertion. Moreover, this mapping language incorporates features dealing with conflicting data. (2) We propose a Bridge Query Language (BQL) that enables a transformation from an SQL query defined over the target to the query executed over a given source. (3) We present a prototype implementation which generates query programs for a popular document oriented database, namely MongoDB, and Cassandra, a column family store.

This paper is organized as follows. In Section 2, we present related works in the domain of querying NoSQL databases. In Section 3, we provide background knowledge on two feature rich and popular NoSQL databases: document and column family stores. Section 4 presents our data integration framework with a presentation of the syntax and semantics of the mapping language. In Section 5, query processing in our data integration system is presented and BQL is detailed. Section 6 concerns aspects of the prototype implementation. Finally, Section 7 concludes this paper.

2 Related Work

In this section, we present some related works in the domain of querying non relational databases in the context of the cloud and Map/Reduce.

Decoupling query semantics from the underlying data store is a widely spread technique to support multiple data sources in one framework. Therefore, various systems offer a common abstraction layer on top of their data storage layer.

Hadoop[8] is a framework that supports data-intensive applications. On top of a distributed, scalable, and portable filesystem (HDFS, [1]), Hadoop provides a column-oriented database called HBase for real-time read and write access to very large datasets.

In order to support queries against these large datasets, a programming model called MapReduce [2] is provided by the system. MapReduce divides workloads into suitable units, which can be distributed over many nodes and therefore can be processed in parallel. However, the advantage of the fast processing of large datasets has also its catch, because writing MapReduce programs is a very time consuming business. There is a lot of overhead even for simple tasks. Working

[8] http://hadoop.apache.org/

out how to fit data processing into the MapReduce pattern can be a challenge. Therefore, Hadoop offers three different abstraction layers for its MapReduce implementation, called Hive, Pig and Cascading.

Hive[9] is a data warehouse infrastructure, which aims to bridge the gap between SQL and MapReduce queries. Therefore, it provides its own SQL like query language called HiveQL [9]. It has traditional SQL constructs like joins, GROUP BY, WHERE, SELECT and FROM clauses. These commands are translated into MapReduce functions afterwards. Hive insists that all data has to be stored in tables, with a schema under its management. Hive allows traditional MapReduce programs to be able to plug in their own mappers and reducers to do more sophisticated analysis.

Like Hive, Pig[10] tries to raise the level of abstraction for processing large data sets with Hadoop's MapReduce implementation. The Pig platform consists of a high level language called Pig Latin [6] for constructing data pipelines, where operations on an input relation are executed one after the other. These Pig Latin data pipelines are translated into a sequence of MapReduce jobs by a compiler, which is also included in the Pig framework.

Cascading[11] is an API for data processing on Hadoop clusters. It is not a new text based query syntax like Pig or another complex system that must be installed and maintained like Hive. Cascading offers a collection of operations like functions, filters and aggregators, which can be wired together into complex, scale-free and fault tolerant data processing workflows as opposed to directly implementing MapReduce algorithms.

In contrast to missing standards in a query language for NoSQL databases, standards for persisting java objects already exist. With the Java Persistence API (JPA[12]) and Java Domain Objects (JDO[13]) it is possible to map java objects into different databases. The Datanucleus implementation of these two standards provides a mapping layer on top of HBase, BigTable [1], Amazon S3[14], MongoDB and Cassandra. Googles App Engine[15] uses this framework for persistence.

A powerful data query and administration tool which is used extensively within the Oracle community is Quest Softwares Toad[16]. Since 2010, a prototype which also offers its support for column family stores is available. During the time of writing this paper, the beta version 1.2 can be connected to Azure Table Services[17], Cassandra, SimpleDB[18], HBase and every ODBC compliant relational database.

[9] http://hive.apache.org/
[10] http://pig.apache.org
[11] http://www.cascading.org/
[12] http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
[13] http://www.oracle.com/technetwork/java/index-jsp-135919.html
[14] http://aws.amazon.com/de/s3/
[15] http://code.google.com/intl/de-DE/appengine/
[16] http://toadforcloud.com
[17] http://msdn.microsoft.com/en-us/library/dd179423.aspx
[18] http://aws.amazon.com/de/simpledb/

Toad for Cloud consists of two components. The first is the Toad client, which can be installed on a Microsoft Windows computer. It can be used to access different databases, the Amazon EC2 console, and to write and execute SQL statements. The second component is the Data Hub. It translates SQL statements submitted through the Toad client into a language understood by all supported databases and returns results in the familiar tabular row and column format.

In order to use SQL on column family stores like Cassandra and HBase, the column families, rows and columns have to be mapped into virtual tables. Afterwards, the user does have full MySQL support on these data, containing also inserts, updates and deletes. Furthermore, it is possible to do virtual data integration with different data sources.

One reason why only column family stores are supported by Toad is their easy mapping to relational databases. To do the same with a document store containing objects with a deep nested structure or squeezing a graph into a relational schema is a much more complicated task. Even if a suitable solution was found, powerful and easy to use query languages, tools and interfaces like Traverser API for Neo4J would be missing in the SQL layer of Toad, which queries the mapped tables.

Due to their different data models and their relatively young history, NoSQL databases still lack a common query language. However, Quest Software and Hadoop demonstrate that it is possible to use SQL (Toad) or a SQL like query language (Hive) on top of column family stores. A mapping to document stores and graph databases is still missing.

3 Background

We have seen that each category of NoSQL databases has its own data model. In this section, we present details concerning document oriented and column family categories.

3.1 Document Oriented Databases

Document oriented databases correspond to an extension of the well-known key-value concept where in this case the value consists of a structured document. A document contains hierarchically organized data similar to XML and JSON. This permits to represent one-to-one as well as one-to-many relationships in a single document. Therefore a complex document can be retrieved or stored without using joins. Since document oriented databases are aware of stored data, it enables to define document field indexes as well as to propose advanced query features. The most popular document oriented databases are MongoDB (10gen) and CouchDB (Apache).

Example 1. In the following a document oriented database stores drug information aimed at an application targeting the general public. According to features proposed by the application, two so-called collections are defined: `drugD` and `therapD`. `drugD` includes documents describing drug related information whereas

`therapD` contains documents with information about therapeutic classes and drugs used for it. Each `drugD` document is identified by a drug identifier. In this example, its attributes are limited to the name of the product, its price, pharmaceutical lab and a list of therapeutic classes. The key for `therapD` documents is a string corresponding to the therapeutic class name. It contains a single attribute corresponding to the list of drug identifiers treating this therapeutic class. Figure 2 presents an extract of this database. Finally, in order to ensure an efficient search to patients an index on the attribute `name` of the `drugD` document is defined.

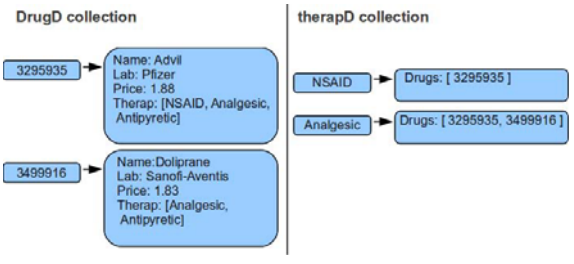


Fig. 2. Extract of drug document oriented database

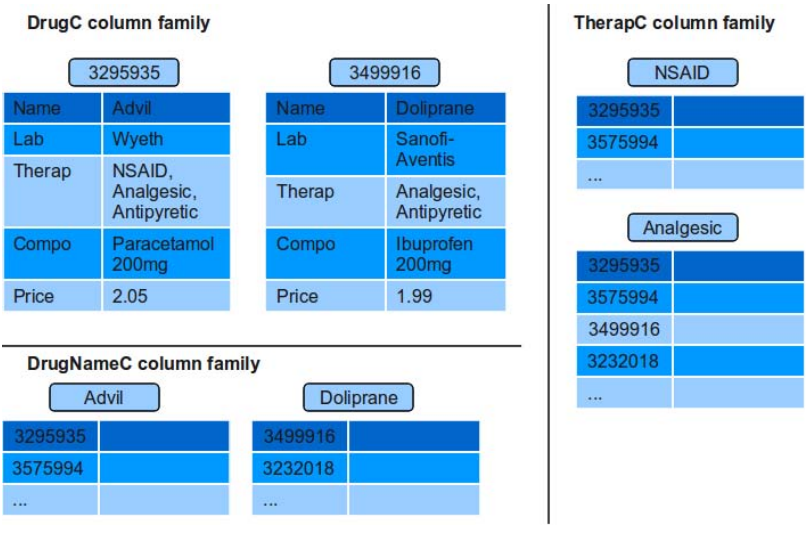


Fig. 3. Extract of drug column family database

3.2 Column-Family Databases

Column family stores correspond to persistent, sparse, distributed multilevel hash maps. In column family stores, arbitrary keys (rows) are applied to arbitrary

key value pairs (columns). These columns can be extended with further arbitrary key value pairs. Afterwards, these key value pair lists can be organized into column families and keyspaces. Finally, column-family stores can appear in a very similar shape to relational databases on the surface. The most popular systems are HBase and Cassandra. All of them are influenced by Googles Bigtable.

Example 2. Figure 3 presents some column families defined in a medical application. Since Cassandra works best when its data model is denormalized, the data is divided on three column families: `drugC`, `drugNameC` and `therapC`. The columns `drugName`, `contra`, `composition`, `lab` are integrated into `drugC` and identified by row key `drugId`. `drugNameC` contains row key `drugName` and a `drugId` column in order to provide an efficient search for patients. Since end-users of this database need to search products by therapeutical classes, `therapC` contains `therapName` as row key and a column for each `drugId` with a timestamp as value.

4 Data Integration Framework

This section presents the syntax and semantics of our data integration framework. Moreover, it focuses on the mapping language which supports the definition of correspondences between sources and target entities. Our mapping language integrates some original aspects by considering (i) query processing performances of the sources via access paths and (ii) dealing with contradicting information found between sources using preferences. In the rest of this paper, we consider the following example.

Example 3. A medical application needs to integrate drug data coming from two different NoSQL stores. The first database, corresponding to a document store, denoted `docDB`, and is used in a patient oriented application while the other database, a column family store, denoted `colDB`, contains information aimed at health care professionals. In this paper, we concentrate on some extracts of `docDB` and `colDB` which correspond to respectively Figures 2 and 3. The data stored in both databases present some overlapping as well as some discrepancies both at the tuple and schema level. For instance, at the schema level, both databases contain french drug identifiers, names, pharmaceutical companies and prices but only `colDB` proposes access to the composition of a drug product. Considering the tuple level, some drugs may be present in one database but not in the other. Moreover information concerning the same drug product (i.e. identified by the same identifier value) may contradict themselves in different sources. Given these source databases, the target schema is defined as follows. We consider that relation and attribute names are self-explanatory.

```
drug(drugId, drugName, lab, composition, price)
therapDrug (drugId, therapeuticName)
```

Obviously, our next step it to define correspondances between the sources and the target. This is supporteed by mapping assertions which are currently being defined manually by domain experts. In the near future, we aim to discover some

of them automatically by analyzing extensions and intensions of both sources and the target. Nevertheless, we do not believe that all mapping assertions can be discovered automatically due to the lack of semantics contained in both the target and the sources. Next, we present the mapping language enabling the definitions of mapping assertions.

4.1 Mapping Language

Our data integration system takes the form of a triple $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ where $\mathcal{T}$ is the target schema, $\mathcal{S}$ is the source schema and $\mathcal{M}$ is the mapping between $\mathcal{T}$ and $\mathcal{S}$. In Section 1, we motivated the fact that the set $\mathcal{S}$ could correspond to both RDBMS and NoSQL stores and that $\mathcal{T}$ takes the form of a relational schema. We consider that the target schema is given, possibly defined by a team of domain experts or using schema matching techniques [7].

This mapping language adopts a GAV (Global As View) approach with sound sources [5]. The mapping assertions are thus of the following form: $\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{T}}$ where $\phi_{\mathcal{S}}$ is a query over $\mathcal{S}$ and $\phi_{\mathcal{T}}$ is a relation of $\mathcal{T}$.

Our system must deal with the heterogeneity of the sources and the highly denormalized aspect of NoSQL database instances. In order to cope with this last aspect, our mapping language handles the different access paths proposed by a given source. This is due to the important performance differences one can observe between the processing of the same query through different access paths. For instance, in the context of `colDB`, retrieving the drug information from a drug name will be more effective using the `drugCName` column family rather than the `drugC` column family (which would require complete scan of all its tuples).

We believe that a mapping assertion corresponds to the ideal place to store the preferred access paths possible for a target relation. Hence each mapping assertion is associated with a list of attributes contained in its target relation. The mapping language enables the use of the '*' symbol which, like in SQL, denotes the complete list of attributes of a given relation. For a given mapping assertion, an access path with attribute 'a' is defined when the source entity offers an efficient access, either using a key or an index, to a collection, set of columns or tuple. Note that for a source corresponding to an RDBMS, the definition of access paths is not necessary since computing the most effective query execution plan will be performed by the system. Hence, definitions of access paths are mandatory only for mapping assertions whose right hand side corresponds to a NoSQL database.

Definition 1. General syntax of a mapping assertion with an access path specification on attribute 'a': `RelationT(a,b,c)` $\overleftarrow{a}$ `EntityS`$(< key; value >)$ where `RelationT` and `EntityS` respectively denote a relation of the target and a conjunction of collections, column families or relations of a source. In this mapping assertion, the attributes of `RelationT` follow the definition order of this relation. Due to the schema flexibility of NoSQL databases, we can not rely upon any attribute ordering in a collection or column family. Hence, we must use attribute names to identify distinct portions of a tuple. In order to map `RelationT` and

`EntityS` attributes, we introduce a '`AS`' keyword to define a correspondence between attribute symbols of the mapping assertion. Finally, an entry in `EntityS` is defined as a key/value structure using a '`<key ; value>`' syntax, where `key` is either (i) '`PKEY AS k`' or (ii) a variable name (previously defined in a `EntityS` couple of the same mapping) and `value` is either of the form (i) `nameS AS nameT` (where nameS and nameT are resp. attribute names from the source and the target) or (ii) of the form `FOREACH item AS name IN list` (where `item` corresponds to an element of the set denoted by `list` and `name` is an attribute identifier of the source). Finally, a keyword is introduced to denote the primary key of the structure (i.e. '`PKEY AS`') and to manipulate it, e.g. `IN KEY`.□

Note the possibility that some target relation attributes are never associated to a mapping assertion. This means that there is not an efficient way to filter a query by this attribute due to the denormalization of the source databases. For instance, a query searching for a given drug composition will be highly inefficient in the `colDB` database due to the absence of a predefined structure proposing a key-based access from this attribute. Finally, for a given source and target relation, there must be a single mapping assertion with a given attribute.

A second feature of our mapping language consists in handling the data structures of NoSQL databases that can be multivalued, nested and also contain some valuable information in the key of a key/value structure; e.g. in the `DrugNameC` column family of Figure 3, drug identifiers of a drug product are stored in the key position (i.e. left hand side of the record). A multivalued example is present in both `docDB` and `colDB` extracts for the `therap` attribute. This forces our mapping language to handle access to the information associated to these constructors, e.g. to enable iteration over lists. Nested attributes are handled by using the standard '.' notation found in object oriented programming. On the second hand, iterations over lists require the introduction of a '`FOREACH`' construct.

We now present the mapping assertions of our running example (a.p. denotes an access path):

1. $drug(i,l,n,c,p)$ $\overleftarrow{*}$ $drugD$(<PKEY AS i ; name AS n, lab AS l, price AS p>)
2. $drug(i,n,l,c,p)$ $\overleftarrow{i}$ $drugC$(<PKEY AS i ; name AS n lab AS l, compo AS c, price AS p>)
3. $drug(i,n,l,c,p)$ $\overleftarrow{n}$ $drugNameC$(<PKEY AS n ; FOREACH id AS i IN KEY>, $drugC$(<id ; lab AS l, compo AS c, price AS p>)
4. $therapDrug(i,t)$ $\overleftarrow{i}$ $drugD$(<PKEY AS i ; FOREACH the AS t IN Therap>)
5. $therapDrug(i,t)$ $\overleftarrow{t}$ $therapD$(<PKEY AS t ; FOREACH id AS i IN Drugs>)
6. $therapDrug(i,t)$ $\overleftarrow{i}$ $DrugC$(<PKEY AS i ; FOREACH the AS t IN Therap>)
7. $therapDrug(i,t)$ $\overleftarrow{t}$ $therapC$(<PKEY AS t ; FOREACH id AS i IN KEY>

This set of mapping assertions examplifies an important part of our mapping language features:

– assertion #1 has a '*' access path since we consider that all attributes of the `drugC` collection are indexed. Also note that on this mapping assertion, the

`c` attribute is not mapped to any source attribute since that information is not available in the `docDB` database.
- Mapping assertion #3 introduces the fact that several source entities can be used in a single mapping (i.e. `drugNameC` and `drugC` column families). Intuitively, this query accesses a given `drugNameC` column family entry identified by a drug name and iterates over its drug identifiers, which are keys (using the '`IN KEY`' expression) then it uses these identifiers to access entries in to `drugC` column family (using `i` iterator variable in the key position of the `drugC`).

4.2 Dealing with Conflicting Data Using Attribute Preferences

In general, data integration and data exchange solutions adopt the certain answers semantics for query answering, i.e. results of a query expressed over the target contain the intersection of data retrieved from the sources. We believe that this pessismistic approach is too restrictive and as a consequence, many valid results may be missing from final results.

At the other extreme of the query answering semantics spectrum, we find the possible answer semantics which provides as results over a target query the union of sources results. With this optimistic approach conflicting results may be proposed as a final result, leaving the end-users unsatisfied.

In this work, we propose a trade-off between these two semantics which is based on a preference-based approach. Intuitively, preferences provided over target attributes define a partial order over mapped sources. Hence, for a given data object, conflicting information on the same attribute among different data sources can be handled efficiently and the final result will contain the preferred values.

Example 4. Consider the queries over `docDB` and `colDB` asking for `lab` and `price` information for the drug identified by value 3295935. Given the information stored in both sources, respectively the column store (Figure 2) and column family store (Figure 3), conflicts arise on the prices, resp. 1.88 and 2.05 euros, and pharmaceuticals, resp. Pfizer and Wyeth.

When creating the mapping assertions, domain experts can express that drug prices are more accurate in the document store (`docDB`) and that information about pharmaceutical laboratory is more trustable in the column family (`colDB`). Hence the result of this query will contain a single tuple consisting of: {`Advil, Wyeth, 1.88`}, i.e. mixing values retrieved the different sources.

We now define the notion of preferences over mapping assertions.

Definition 2. Consider a set of source databases $\{DB_1, DB_2, ..DB_n\}$, a preference relation, denoted $\succ$, is a relation $\succ \subseteq DB_i \times DB_j$, with $i \neq j$, that is defined on each non primary key attribute of target relations. A preference $\succ$ is total on an attribute A if for every pair $\{DB_i, DB_j\}$ of sources that propose attribute A, either $DB_i \succ^* DB_j$ or $DB_j \succ^* DB_i$ with $\succ^*$ the transitive closure of $\succ$. □

Example 5. Consider the `drug` relation in our running example target schema. Its definition according to the preferences proposed in Example 3 are the following: `drug(drugId, drugName`$^{docDB \succ colDB}$`, lab`$^{colDB \succ docDB}$`, composition, price`$^{docDB \succ colDB}$`)`

That is, for a given drug, in case of conflict, its `docDB` `drugName` attribute is preferred to the one proposed by `colDB` and the preferred value for the `lab` attribute is `colDB` over `docDB`. Note that since the composition attribute can only be retrieved from the `colDB` source, it is not necessary to define a preference order over this attribute.

5 Query Processing

Once a target relation schema and a set of mapping assertions have been defined, end-users can expressed queries in SQL over the target database. Since that database is virtual, i.e. it does not contain any data, data needs to be retrieved from the sources and processed to provide a final result. The presence of NoSQL databases in the set of sources imposes to transform the former SQL query into a query specifically tailored to each NoSQL source. This transformation is based on the peculiarities of the source database, e.g. whether a declarative query language exists or only procedural approach enables to query that database, and the mapping assertions. Since most NoSQL stores support only a procedural query approach, we have decided to implement a query language to bridge the gap between SQL and some code in a programming language. This section presents the Bridge Query Language (henceforth BQL) which is used internally by our data integration system and the query processing semantics.

5.1 Architecture

The overall architecture of query processing within our data integration system is presented in Figure 4. First, an end-user writes an SQL query over the target schema. The expressivity of accepted SQL queries corresponds to `Select Project Join` (SPJ) conjunctive queries, e.g. `GROUP BY` clauses are not accepted but we are planning to introduce them in future extensions. Note that this limitation is due to a common abstraction of the NoSQL databases we are studying in this paper (column family and document).

An end-user SQL query is then translated into the BQL internal query language of our data integration system. This transformation corresponds to a rewritting of the SQL into a BQL query using the mapping assertions. Note that this translation step is not needed for a RDBMS.

Then for each BQL, a second transformation is performed, this time to generate a query tailoring the NoSQL database system. Thus, for each supported NoSQL implementation, a set of rules is defined for the translation of a BQL query. Most of the time, the BQL translation takes the form of a program and uses a specific API. In Section 6, we provide details on the translation from BQL to Java programs into MongoDB and Cassandra.

The results obtained from each query is later processed within the data integration system. Intuitively, each result set takes the form of a list containing the awaited target columns. In order to detect data conflicts, we need to efficiently identify similar objects. This step is performed by incorporating into the result set values corresponding to primary keys of target relations of the SQL query. So, even if primary keys are not supposed to be displayed in the final query result, they are temporarily stored in the result set. Hence objects returned from the union of the result sets are easily and unambiguously identified. Similar objects can then be analyzed using the preference orders defined over target attributes. The query result contains values retrieved from the preferred source attributes.

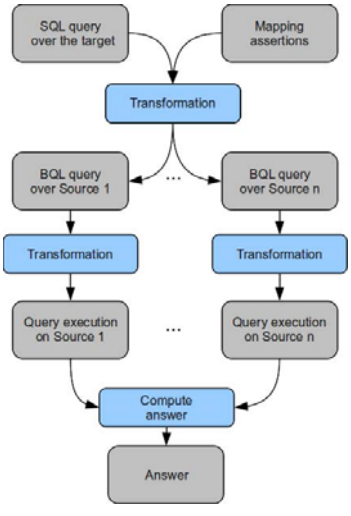


Fig. 4. Query processing

5.2 Bridge Query Language

BQL is the internal query language that bridges the gap between the SQL language of the target model and the different and the heterogenous query languages of the sources. The syntax of the query language follows the EBNF proposed in the companion web site. This language contains a set of reserved words whose semantics is obvious for a programmer. For instance, the `get` instruction enables to define a set of filter operations and to define the distinguished variables of the query, i.e. the values needed in the result. The `foreach in :` instruction is frequently encountered in many programming languages and their semantics align. Intuitively, it supports an iteration over elements of a result set and the associated processing is performed after the `':'` symbol. We have implemented an SQL to BQL translator which parses an SQL query and generates a set of

BQL queries, one for each NoSQL database mapped to the relation of the target query. This translator takes into account the mapping assertions defined over the data integration system.

Example 6. We now introduce a set of queries expressed over our running example. They correspond to different real case scenario and emphasize the different functionalities of our query language. For each target query (SQL), we present the BQL generated for both `colDB` and `docDB`.

- Query 1 accesses a single table via its primary key.
  SQL: `SELECT drugName, price FROM drug WHERE drugId=3295935;`
  docDB's BQL: `ans(drugName, price) = docDB.drugD.get({PKEY= 3295935},{name, price})` provides answer (Advil, 1.88)
  colDB's BQL: `ans(drugName, price) = colDB.drugC.get({PKEY= 3295935},{name, price})` provides answer (Advil, 2.05)
  Answer: Since the query identifies tuples with the primary key, the real world object of the answers is supposed to be the same and we apply the preferences over the union of the results. The processed result is (Advil, 1.88)
- Query 2 access single table over a non primary key but indexed attribute of the target.
  SQL:`SELECT drugId, price FROM drug WHERE drugName LIKE'Advil';`
  docDB's BQL: `ans(drugId, price) = docDB.drugD.get({name='Advil'}, {PKEY, price})` with answer {(3295935, 1.88)}
  colDB's BQL: `temp(drugId) = colDB.drugNameC.get({name='Advil'}, {KEY})`
  `ans(drugId, price)=foreach id in temp(drugId):colDB.drugC.get( {KEY=id},{KEY,price}) colDB.drugC` with {(3295935, 2.05),(3575994, 2.98)} Answer: The final result set is {(3295935, 1.88),(3575994, 2.98)} thus mixing the results and taking advantage of the preference setting.
- Query 3 retrieves data from a single relation with a filter over a non-primary and non-indexed attribute of the target.
  SQL: `SELECT drugName FROM drug WHERE lab='Bayer';`
  docDB's BQL: `ans(drugName)=docDB.drugD.get({lab='Bayer'},{name})`
  colDB's BQL: No solution
  Answer: Since no queries are generated for the `colDB` store, the results are retrieved solely from `docDB`.
- Query 4 involves 2 relations and an access from a single primary key attribute of the target.
  SQL:`SELECT drugName FROM drug d, therapDrug td WHERE d.drugId=td.drugId AND therapId LIKE 'NSAID';`
  docDB's BQL: `temp(drugs) = docDB.therapD.get({PKEY='NSAID'},{drugs})`
  `ans(drugName) = foreach id in temp(drugs) : docDB.drugC.get( {PKEY=id},{name})`
  colDB's BQL: `temp(drugs) = colDB.therapC.get({PKEY='NSAID'},{KEY})`
  `ans(drugName)=foreach id in temp(drugs) : coldDB.drugC.get({PKEY=id},{name})`
  Answer: provides the same result as in Query 2.

6 Implementation

In this section, we sketch our prototype implementation which tackles a document store (MongoDB) and a column store (Cassandra). Together they represent some of the most popular open source projects in the NoSQL ecosystem. The platform we have adopted corresponds to Java since both MongoDB and Cassandra propose APIs and enable the execution of this programming language. Moreover, Java is adapted to numerous other NoSQL stores. Nevertheless, in the near future, we are planning to tackle other systems, e.g. CouchDB or HBase, and consider other access methods, e.g. javascript or python.

An important task of our prototype is to handle the transformation modules found in Figure 4. That is to process the translation (i) from SQL to BQL and (ii) from BQL to the query language supported by each NoSQL stores. Due to the declarative nature of both query languages of the former translation, this task is easy to implement and is implemented in linear time on the length of the input SQL query. The latter translation task is more involved since BQL corresponds to a declarative language and the target query of our NoSQL stores corresponds Java methods. The high denormalization aspect of NoSQL stores imposes that only a limited set of queries can be efficiently processed. In fact, this results in having similarities between queries expressed over a given NoSQL database instance. We have extract these similarities into patterns which are implemented using Java methods. The main idea is to consider a set of finite BQL templates and to associate a Java method to each of these templates.

One can ask the following question: is this approach still valid and efficient when more complicated SQL queries, e.g. involving aggregate functions (min, max, etc.), group by and having or like constructors? A reply to this question necessarily needs to consider the particular features of each NoSQL database supported by the system since, up to now, a common framework for NoSQL stores does not exist. In the case of MongoDB, the support of regular expressions enables the execution of complex statements with minimal additional effort. On the other hand, Cassandra only supports a range query approach on primary keys. This results in having an inefficient support for aggregate operations and queries involving regular expressions.

7 Conclusions

This paper is a first approach to integrate data coming from NoSQL stores and relational databases into a single virtualized database. Due to the increasing popularity of this novel trend of databases, we consider that such data integration systems will be quite useful in the near future. Our system adopts a relational approach for the target schema which enables end-users to express queries in the declarative SQL language. Several transformation steps are then required to obtain results from the data stored at each of the sources. Hence a bridge query language has been presented as the cornerstone of these transformations. Another important component of our system is the mapping language which

(i) handles uncertainty and contradicting information at the sources by defining preferences over mapping assertions and (ii) supports the setting of access path information in order to generate an efficiently processable query plan.

On preliminary results, the overhead of these transformation steps does not impact the performance of query answering. Our list of future works is important and among others, it contains the support of NoSQL stores corresponding to a graph model and the (semi) automatic discovery of mapping assertions based on the analysis of value stored in each source.

References

1. 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 (awarded best paper!). In: OSDI, pp. 205–218 (2006)
2. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)
3. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. In: SOSP, pp. 205–220 (2007)
4. Kifer, M., Bernstein, A., Lewis, P.M.: Database Systems: An Application Oriented Approach, Complete Version, 2nd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2005)
5. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS, pp. 233–246 (2002)
6. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig latin: a not-so-foreign language for data processing. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1099–1110. ACM, New York (2008)
7. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10, 334–350 (2001)
8. Stonebraker, M., Madden, S., Abadi, D.J., Harizopoulos, S., Hachem, N., Helland, P.: The end of an architectural era (it's time for a complete rewrite). In: VLDB, pp. 1150–1160 (2007)
9. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive - a warehousing solution over a map-reduce framework. PVLDB 2(2), 1626–1629 (2009)

An Energy-Efficient Concurrency Control Algorithm for Mobile Ad-Hoc Network Databases

Zhaowen Xing and Le Gruenwald

School of Computer Science
University of Oklahoma
Norman, OK 73019, USA
{zhaowenxing,ggruenwald}@ou.edu

Abstract. MANET does not require any fixed infrastructure, thus it fits well in disaster rescue and military operations. However, when a node has no or insufficient energy to function, communication may fail, disconnections may happen, and transactions may be aborted if they are time-critical and miss their deadlines. Energy-efficient transaction management becomes an important issue in MANET database applications. In this paper, we propose an energy-efficient concurrency control (CC) algorithm for MANET databases in a clustered network architecture where nodes are divided into clusters, each of which has a node, called cluster head, responsible for the processing of all nodes in the cluster. In our algorithm, in order to conserve energy and balance the energy consumption among servers, we elect cluster heads to work as coordinating servers, and propose an optimistic CC algorithm to offer high concurrency and avoid wasting limited system resources. The simulation results confirm that our technique performs better than existing techniques.

Keywords: Mobile ad-hoc network, clustering, transaction management, concurrency control.

1 Introduction

A Mobile Ad-hoc Network (MANET) is a collection of mobile, wireless and battery-powered nodes, and every node can roam freely. A mobile database system built on a MANET is called a MANET database system. In this system, both clients and servers are mobile, wireless and battery-powered, and the databases are stored at servers and accessed by clients. As no fixed infrastructure is required, MANET databases can be deployed in a short time and mobile users can access and manipulate data anytime and anywhere, and they become an attractive solution for handling mission-critical database applications, such as disaster response and recovery systems [1, 2, 3] and military operations like battlefields [1]. In these applications, transactions must be executed not only correctly but also within their deadlines. To guarantee this, a concurrency control (CC) technique must be a part of the system.

CC is the activity of preventing transactions from destroying the consistency of the database while allowing them to run concurrently, so that the throughput and resource utilization of the database system are improved and the waiting time of concurrent

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 496–510, 2011.

transactions is reduced [4]. However, because of their mobility and portability, mobile nodes have severe resource constraints in terms of battery capacity, memory size and CPU speed. As the battery capacity is limited, it compromises the ability of each mobile node to support services and applications [5]. Also battery technology is not developed as rapidly as mobile devices and wireless technologies, so that the limited battery lifetime is always a bottleneck for the development of improved mobile devices [6]. Therefore, a suitable CC algorithm for MANET databases should be energy-efficient.

Although there are CC algorithms proposed for cellular mobile network databases [7, 8, 9], to the best of our knowledge, only one CC algorithm has been proposed for MANET databases [10]; however, this algorithm not only relaxes transaction atomicity and global serializability, but also does not take energy efficiency into account. To fill this gap, in this paper we propose an optimistic CC algorithm, called Sequential Order with Dynamic Adjustment (SODA), which takes mobility, real-time constraints and energy efficiency into consideration, to handle mission-critical databases in a clustered MANET architecture. Our objective is to minimize energy consumption of each mobile node, and balance energy consumption among servers, so that mobile nodes with low energy do not run out of energy quickly, and thus, the number of disconnections and transaction aborts due to low energy or energy exhaustion can be reduced. The rest of the paper is organized as follows. Section 2 reviews some of recent CC algorithms for mobile databases. Section 3 describes the proposed MANET architecture. Section 4 presents our CC algorithm, SODA. Section 5 provides the simulation results. Finally Section 6 concludes the paper with future research.

2 Related Work

As cellular mobile networks and MANET have many similar characteristics, in this section, we review the CC techniques recently proposed for databases in both networks.

Look-Ahead Protocol (LAP) was proposed in [8] to maintain data consistency of broadcast data in mobile environments. In LAP, update transactions are classified into either hopeful or hopeless transactions. Hopeless transactions can not commit before their deadlines and are aborted as early as possible to save system resources and reduce data locks, while hopeful transactions can continue to execute their read and write operations using the two-phase locking (2PL) algorithm.

Multi-Version Optimistic Concurrency Control for Nested Transactions (MVOCC-NT) [9] was proposed to process mobile real-time nested transactions using multi-versions of data in mobile broadcast environments. MVOCC-NT adopts the timestamp interval with dynamic adjustment to avoid unnecessary aborts. At mobile clients, all active transactions perform backward pre-validation against transactions committed in the last broadcast cycle at the fixed server. Read-only transactions can commit locally if they pass the pre-validation, but surviving update transactions have to be transferred to the fixed server for the final validation. Choi et al. [7] proposed 2-Phase Optimistic Concurrency Control (2POCC) to process mobile transactions in wireless broadcast environments. Transaction validation is done in two phases: partial

backward validation at mobile clients and final validation at the fixed server. In both phases, if a transaction T_i is serialized before transaction T_j then the writes of T_i should not overwrite the writes of T_j and affect the reads of T_j.

All the CC techniques reviewed above were designed for cellular mobile databases, which heavily rely on broadcast techniques to save mobile nodes' energy and on static servers that have no energy limitation; thus, they are not suitable for MANET databases.

Semantic Serializability Applied to Mobility (SESAMO) [10] was proposed for MANET databases. SESAMO is based on semantic serializability, which requires that not only databases on mobile nodes be disjoint but also updates on a database depend only on the values of the data in the same database; therefore, transaction atomicity and global serializability can be relaxed. However, in SESAMO, global transactions still need be serialized at each site using strict two-phase locking (S2PL), while at the same time each site must maintain the consistency of its own local database. SESAMO does not take energy efficiency into account. In addition, in MANET databases for mission-critical applications, the assumption for semantic serializability does not hold because each database depends on each other due to the organizational structure of the applications. For example, in a disaster rescue scenario, before sending firefighters out to pursue some actions, the status of their equipment has to be checked, where the firefighter database may be stored on one mobile server, and the equipment database may be stored on another mobile server.

3 Proposed Architecture

In this section, we introduce our clustered MANET architecture which is built by applying our robust weighted clustering algorithm, called MEW (Mobility, Energy, and Workload) [11]. MEW takes mobile nodes' mobility, energy and workload into consideration when grouping mobile nodes into clusters a MANET. In this architecture, mobile nodes are divided into clusters, each of which has one cluster head working as the coordinating server responsible for the transaction processing of the mobile nodes, called cluster members, within the cluster. Cluster heads can communicate with each other through some mobile nodes that work as gateways. Similarly, mobile nodes from different clusters can also communicate with each other, but they have to go through their cluster heads to get the destination addresses first. Also, nodes are put into the same cluster based on their application semantics.

We choose this architecture for three reasons. First, in many MANET applications, such as disaster response and recovery systems [2, 3] and military operations [12], users are logically grouped. Second, because every node is mobile in a MANET, the network topology may change rapidly and unpredictably over time. According to [13], clustered architectures are proper to keep the network topology stable as long as possible, so that the performance of routing and resource relocation protocols is not compromised. Third, in order to accommodate our optimistic CC algorithm SODA to guarantee global serializability, the information of committed global transactions is maintained by the cluster head with the highest remaining energy.

Fig. 1 shows an example of a clustered MANET database architecture with three clusters, each of which is represented by a large solid circle with mobile clients and servers shown as PDA/iphone and laptop icons, respectively. The arrows between the devices show the communication between them. In the rest of this section, we describe the functionality of mobile nodes (Section 3.1), the MEW algorithm (Section 3.2), the cluster formation (Section 3.3), and the cluster maintenance (Section 3.4).

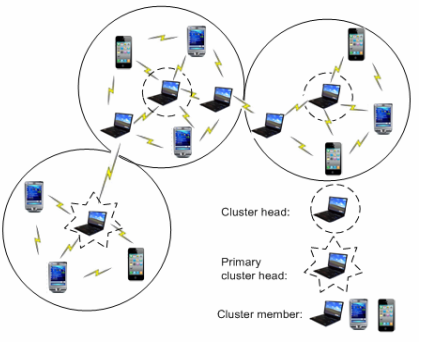


Fig. 1. Architecture of a clustered MANET database

3.1 Mobile Node Functionality

Depending on the communication strength, computing power and storage size, mobile nodes are classified into clients and servers. On clients, only the query processing modules that allow them to submit transactions and receive results are installed, while on servers, the complete database management systems are installed to provide transaction processing services. Servers are further classified into coordinating servers and participating servers. Coordinating servers are the ones which receive global transactions from clients, divide them into sub-transactions, forward these sub-transactions to appropriate participating servers, and maintain the ACID (Atomicity, Consistency, Isolation, and Durability) [4] properties of global transactions. Participating servers are the ones that process sub-transactions transmitted from coordinating servers, and preserve their ACID properties.

The entire database is partitioned into local databases and distributed to different servers, and there is no caching or replication technique involved for simplicity. Transactions are based on the simple flat model, which contains a set of read, write, insert, and delete operations. Any subset of operations of a global transaction that access the same server is submitted and executed as a single sub-transaction.

With respect to the clustered MANET architecture shown in Fig. 1, a mobile node is either a cluster head when it is a coordinating server or is a cluster member when it is either a participating server or a client. In order to guarantee global serializability and reduce communication overhead, among cluster heads the one with highest remaining energy is further elected as the primary cluster head, which maintains the information of committed global transactions and validates transactions globally.

3.2 The Basis of Our Clustering Algorithm - MEW

Being inspired by the mobility based clustering algorithm, MOBIC [14], and the weighted clustering algorithm, WCA [13], and considering a new system parameter "Energy Decreasing Rate" (*EDR*), we propose a weighted clustering algorithm, called MEW (Mobility, Energy, and Workload), to build a stable backbone in MANETs. The objective of MEW is to form and maintain stable clusters in MANETs by electing nodes with the highest weights as cluster heads, where the weight of a node is calculated as a combination of its mobility, energy and workload.

To capture the mobility of nodes, we do not consider their absolute roaming speed. This is because it is easy to calculate the speed's quantity, but it is hard to predict the direction of movement. Without the direction, the speed's quantity alone is not appropriate to justify whether a node is a good candidate for cluster head or not. For instance, two nodes that have small speeds move in the opposite directions. As time goes, they will be out of each other's transmission range and get disconnected from each other. Also the utilization of GPS is opted out because GPS consumes the limited battery energy of mobile nodes.

Instead, two mobility metrics, Relative Mobility between two nodes i and j (RM_{ij}) [14] and Mobility Prediction for node j (MP_j), are introduced to monitor the mobility of nodes and applied to determine whether a node is suitable to be a cluster head. RM_{ij} measures whether node i and node j move relatively together; MP_j measures whether all 1-hop neighbors of node j move relatively together along with node j. Below we explain how each of the two metrics is calculated.

For each node j ($1 \le j \le N$ for N nodes in the network), after receiving two successive HELLO messages from every 1- hop neighbor i ($1 \le i \le n$ if there are n neighbors), the RM_{ij} is calculated by Equation (1). RSS_{ij1} and RSS_{ij2} are the received signal strengths (*RSS*) that are read from the *RSS* indicator when the first and second HELLO messages from the same neighbor i are received by node j, respectively. Based on the value of RM_{ij}, we can say that if RM_{ij} is equal to 1, then the node j and its neighbor i either do not move at all or move with the same speed in the same direction; if RM_{ij} is less than 1, then they move close to each other; otherwise, they move away from each other.

$$RM_{ij} = \frac{RSS_{ij1}}{RSS_{ij2}} \tag{1}$$

For each node j, in order to take into account the mobility of all n 1-hop neighbors and have an integral value to represent them, MP_j is calculated as the standard deviation of RM_{1j}, RM_{2j}, …, RM_{nj} shown in Equation (2). However, for the stability of elected clusters, we prefer RM_{ij} to be equal to or less than 1 because we want cluster heads not to move away from their members. Thus, in the MP_j calculation, the mean of RM_{ij} ($1 \le i \le n$) is 1 instead of the actual mean. A node j with a lower MP_j means that it stays closer to its neighbors, thus, it is a better candidate for the cluster head among its neighbors.

$$MP_j = \sqrt{\frac{\sum_{i=1}^{n} (RM_{ij} - \overline{RM_{ij}})^2}{n}}, \text{ where } \overline{RM_{ij}} = 1 \tag{2}$$

When dealing with the limited battery energy, we consider not only the Remaining Energy (*RE*) of each node but also its Energy Decrease Rate (*EDR*) as the workload because nodes with a heavier workload consume more energy, so that we can balance the energy usage and prevent cluster heads from running out of energy quickly. In other words, for each node j, EDR_j is considered because RE_j represents only the current state of the energy level and the node's energy will run out soon if it normally has a heavy workload. The EDR_j at time interval $[t_1, t_2]$ is calculated by using Equation (3), where RE_{j1} and RE_{j2} are the remaining energy of node j at time t_1 and t_2, respectively.

$$EDR_j = \frac{RE_{j_1} - RE_{j_2}}{t_2 - t_1} \tag{3}$$

A node with a lower *EDR* indicates that it was not busy at least during the interval $[t_1, t_2]$. However, when a node has a busy work history, it most likely will be busy in the future as well. Since the larger the time interval is, the more accurate the *EDR* is in indicating a node's workload history, during the initial election, each node saves a copy of its initial remaining energy and initial time as RE_{j1} and t_1, so that a more accurate *EDR* can be calculated in the future cluster head reelection.

Based on the above analysis about mobility, energy and workload, it is obvious that a node *j* is the best candidate for a cluster head among all its neighbors if its RE_j is the highest, its MP_j is the lowest and its EDR_j is the lowest. In other words, a node with the highest weight is the best candidate for a cluster head when we combine these three metrics together as the weight, which is calculated in Equation (4). Since these metrics have different units, we apply the inversed exponential function to normalize MP_j and EDR_j and bound their values between 0 and 1. RE_j is left out because the value of the remaining energy is at most 100%.

$$W_j = f_1 * e^{-MP_j} + f_2 * RE_j + f_3 * e^{-EDR_j} \tag{4}$$

In Equation (4), $RE_j = RE_{j2}$, the weighting factors f_1, f_2 and f_3 are set according to different application scenarios, and $f_1 + f_2 + f_3 = 1$. When we let $f_2 = f_3 = 0$, that is, we take away the effect of energy and workload, our algorithm turns into a mobility-only-based approach just like MOBIC [14].

3.3 Cluster Formation

To form clusters, each node first broadcasts HELLO messages, collects 1-hop neighbors' information, and computes its weight based on mobility, energy and workload using Equations (1), (2), (3) and (4) defined above. After broadcasting their own weights and receiving all 1-hop neighbors' weights, nodes with the highest weights declare themselves as cluster heads, and 1-hop neighbors of cluster heads join them as cluster members. The details of the cluster formation and different types of messages used in cluster formation are presented in [11]. Note that because clients have less communication strength, less computing power and smaller storage size than servers, clients cannot be elected as cluster heads and cannot work as coordinating servers.

3.4 Cluster Maintenance

Because every node can roam and has limited battery power in a MANET, cluster heads can resign due to low remaining energy, the links between cluster members and cluster heads can be broken, and the links between two cluster heads can be generated [15]. Consequently, clusters need be re-clustered. In other words, leaving clusters, joining clusters, merging clusters, and re-electing cluster heads are normal re-clustering operations in a clustered MANET. However, these operations should be performed only on demand to reduce the overhead of computation and communication, and to provide consistent quality of service.

In order to maintain connections with neighbors, detect the link breaks and new link establishments, each node needs periodically broadcast HELLO messages. Being a cluster head, it has to periodically monitor (after a global transaction commits) its remaining energy level so that it will resign from the cluster head status when the remaining energy drops below a predefined Low Energy Threshold (*LET*). Relying on these two periodic operations, cluster maintenance can be done by recovery from a link break between a member and its cluster head, recovery from a link establishment when two cluster heads become 1-hop neighbors, or recovery from a link break when a cluster head resigns because its current remaining energy of a cluster head is less than *LET*. The details of these recovery tasks are discussed in [11].

4 Proposed Concurrency Control Algorithm: SODA

In this section, we describe our proposed CC algorithm, called Sequential Order with Dynamic Adjustment (SODA). We first show how SODA works in a centralized database as originally presented in [17]. We then discuss how SODA works in a clustered MANET database.

In [17], SODA is proposed for mobile P2P databases, in which each peer carries its own local database, is fully autonomous and shares information in on-the-fly fashion. Therefore, although global transactions (or called remote queries) do exist, it is unnecessary to maintain the global serializability among peers, which is required in traditional distributed databases and MANET databases that we focus on in this paper. However, in mobile P2P databases every peer still needs to guarantee the correctness of transactions that it processes locally because it may collect or update its own data, reply to requests and update replica simultaneously.

4.1 How SODA Works in a Centralized Database

Inspired by the dynamic adjustment technique proposed in [18], and based on the combination of Timestamp Ordering (TO), Optimistic Concurrency Control (OCC), and backward validation, we propose an optimistic CC algorithm called SODA. In SODA, a list of committed transactions is maintained to validate committing transactions. During the validation, the list can be dynamically adjusted to avoid unnecessary aborts. After the committing transaction commits, the list is updated and trimmed.

Assume that T_i's (i = 1, …, n) are committed transactions, and T is a validating/committing transaction. If we simply let the validation/commit order be the serialization order like in traditional OCC, and if there is a read-write conflict between T and T_i, i.e., T reads a common data item d before T_i updates d, then T is aborted because two orders are different. Such aborts should be avoided.

To avoid such aborts, in SODA, a dynamic order instead of validation order among committed transactions is used. In SODA, a Sequential Order (SO) of committed transactions is maintained as $\{T_1, T_2, \ldots, T_i, \ldots, T_n\}$ (also called a history list, which is ordered from left to right) and can be dynamically adjusted. The dynamic adjustment consists of simple and complex cases. In the simple case, the validating transaction T can commit if it can be directly inserted into the maintained sequential order without

adjustment, and the final sequential order will be $\{T_1, T_2, \ldots, \boldsymbol{low}, \ldots \mathbf{T}, \boldsymbol{up}, \ldots, T_n\}$, such that T must-be-serialized-after the transaction *low* but before the transaction *up*. On the other hand, in the complex case, the sequential order must be adjusted before the insertion of T. After T passes the validation and commits, the maintained SO is updated with T's information. In addition, old committed transactions are removed from the maintained SO to reduce the overhead and save the limited storage.

To prove the correctness (or completeness) of SODA, we must show that any schedule produced by SODA is serializable. To fulfill this goal, we proved that the new serialization graph is still acyclic after the addition of any newly committed transaction that has passed our validation test. Further details can be found in [17].

4.2 How SODA Works in a Clustered MANET Database

In order to make SODA work effectively in a clustered MANET database, the coordinating server functionality is combined with the cluster head's functionality because a cluster head is elected by our MEW algorithm as described in Section 3 and is the nearest server with the highest energy in clients' neighborhood. This would enable clients to save time, limited battery energy and bandwidth that they must spend on identifying suitable servers to which they send their transactions. Therefore, only three major functionalities are required: the primary cluster head functionality, cluster head functionality, and participating server functionality as shown in Fig. 2. Note that one server can have all the three functionalities at the same time.

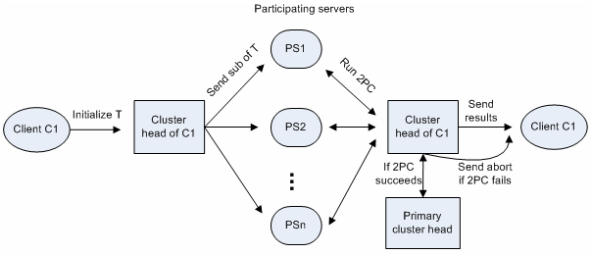


Fig. 2. Transaction flow in a clustered MANET database

4.2.1 Transaction Execution Model

As shown in Fig. 2, a transaction T issued by a client is distributed to its cluster head; the cluster head divides T into sub-transactions and transmits them to the appropriate participating servers according to the global database schema. Each participating server processes the sub-transactions locally and sends the results back to the cluster head. The cluster head runs the 2-Phase Commit (2PC), and gathers all results from the participating servers. Note that we adopt 2PC here due to its simplicity as our research goal is to develop a concurrency control algorithm, not a commit algorithm; however, we do plan to include a more suitable commit protocol for MANET databases in our future work. If running 2PC successfully, the cluster head sends T to the primary cluster head to validate T globally based on the SO of committed global transactions; otherwise, the cluster head sends an abort message directly to the client. After receiving the global validation result, the cluster head sends the final results to the client.

4.2.2 The Primary Cluster Head Functionality

The primary cluster head has the following functionalities:

- It maintains the sequential order (SO) of committed global transactions.
- It receives global transaction validation requests from non-primary cluster heads.
- It validates global transactions using SODA. After validation, it sends the validation results to the non-primary cluster head.
- It updates the SO after a global transaction commits and adds this global transaction's read set, write set and the timestamp of both sets to the data structure of the maintained SO.
- It removes the old committed transactions that are not serialized after any active/committed global transaction from the maintained SO after a global transaction commits.
- It periodically checks (after a global transaction commits) its remaining energy level. If its level is below a predefined threshold *LET* and another cluster head's remaining level is above the threshold, it resigns its cluster head status and elects a new primary cluster head that has the highest remaining energy from all cluster heads. It then transfers the information of all the transactions it stores to the new primary cluster head. Note that since the primary cluster head is also a non-primary one, if the primary one resigns, the non-primary one also resigns if there is a candidate in the neighborhood.

4.2.3 The Non-primary Cluster Head Functionality

A non-primary cluster head has the following functionalities:

- It receives a global transaction from a client, divides them into sub-transactions, and sends the sub-transactions to appropriate participating servers.
- It runs 2PC to request the status of the sub-transactions and requests the timestamps of the global transaction's read set.
- It propagates the global transaction to the primary cluster head after it receives all successful messages of the sub-transactions. After receiving the validation result, it sends the final results to the client.
- It periodically checks (after a global transaction commits) its remaining energy level. If the level is below a predefined threshold and there is a candidate for cluster head in the neighborhood, it resigns its cluster head status and elects a new cluster head in the neighborhood. It then transfers the information of all the transactions it stores to the new cluster head. Note that if the old cluster head is also the primary cluster head, then the new cluster head can be the new primary cluster head as well if this new one has the highest remaining energy among all cluster heads.

4.2.4 The Participating Server Functionality

A participating server has the following functionalities:

- It receives and processes sub-transactions, and maintains the SO of committed sub-transactions.
- It runs SODA locally based on the local SO of committed sub-transactions when it receives the request about the status of the sub-transactions.

- It sends the final status of the sub-transactions to the requesting cluster head. It also sends the timestamps of the read sets of the sub-transactions to the cluster head if the sub-transactions pass the validation.
- It updates the local SO of committed sub-transactions if a sub-transaction commits and adds this sub-transaction's read set, write set and timestamps of both sets to the data structure of the maintained SO. It removes the old committed sub-transactions that are not serialized after any active/committed sub-transaction from the maintained SO after a sub-transaction commits.

5 Performance Evaluation

The simulation experiments are conducted to compare the performance of our proposed SODA with that of SESAMO [10] as unlike other existing CC for cellular mobile databases, SESAMO was specifically designed for MANET databases.

Our simulation models consist of a transaction generator, a real-time scheduler that schedules transactions using early deadline first, participating servers, coordinating servers or cluster heads for SODA only, and a deadlock manager for SESAMO. The simulation model of SODA is the same as the transaction execution model discussed in Section 4.2.1. The simulation model for SESAMO is similar to the one of SODA except for a couple of points. First, SODA is applied locally and globally to validate transactions, while in SESAMO, strict 2PL is applied locally [19] and globally [10]. Second, SESAMO does not elect cluster heads and does not apply 2PC; so it may randomly choose a server as the coordinating server.

5.1 Simulation Parameters and Performance Metrics

Our simulation models are implemented using the AweSim simulation language [20]. Global transactions are defined as entities, and mobile modes are defined as resources with different initial energy levels and randomly distributed locations.

The static simulation parameters and their values are shown in Table 1. We conducted experiments to study the impacts of inter-arrival time on the system performance, which is the mean of an exponentially distributed time between the arrivals of two consecutive transactions. The inter-arrival time is varied over the range from 1 to 10 seconds in order to vary the system load [21] and create a scenario with high data contention.

Table 1. Simulation Parameters

Static Parameters	Value	Reference
Simulation area (m^2)	1000x1000	[21]
Transmission range (m)	250	[22]
Node moving speed (m/s)	2	[22]
Total transactions	1000	[21]
Low Energy Threshold (LET)	50%	
Server energy consumption rate in active mode (watts)	30.3	[23]
Server energy consumption rate in doze mode (watts)	12.5	[23]

Five performance metrics are used and they are defined in Equations (5), (6), (7), (8), and (9), respectively: total time when servers are in active mode, abort rate, total number of cluster head reelections, total energy consumed by all servers, and average difference in remaining energy between two servers. Since transactions in mission-critical applications should be executed not only correctly but also within their deadlines, we use firm real-time transactions to evaluate the performance. In our simulation, a transaction will be aborted if either it misses its deadline or the system could not complete it successfully (e.g. when it is aborted by the CC technique).

A server is in active mode only if it is processing transactions; otherwise, it is in doze mode to save energy. The total time when servers are in active mode evaluates whether servers are busy to process transactions most of time, where m is the total number of servers and $T_{a,i}$ is the total time when server S_i is in active mode.

$$Total\ time\ when\ servers\ are\ in\ active\ mode\ = \sum_{i=1}^{m} T_{a,i} \tag{5}$$

The second performance metric is the abort rate to measure the percentage of aborted transactions, and can be computed as below:

$$Abort\ rate = \frac{Total\ \#\ of\ aborted\ transactions}{Total\ \#\ of\ generated\ transactions} * 100\% \tag{6}$$

The third performance metric is the total number of cluster head (primary and non-primary) reelections to evaluate whether an algorithm takes balancing energy among servers into consideration, where N_{primay} ($N_{non\text{-}primay}$) is the number of primary (non-primary) cluster head reelections. However, more reelections do not guarantee more balanced energy among servers because there is an overhead of reelections and transferring the information from the old cluster head to the new one.

$$Total\ number\ of\ cluster\ head\ reelections\ = N_{primay}\ + N_{non-primary} \tag{7}$$

The fourth performance metric is the total amount of energy consumed by all servers in both active mode and doze mode. This metric evaluates how energy-efficient each technique is, where m is the total number of servers, ECR_a (ECR_d) is the energy consumption rate when a server is in active (doze) mode, and $T_{a,i}$ ($T_{d,i}$) is the total time when server S_i is in active (doze) mode.

$$Total\ energy\ consumed\ by\ servers\ = \sum_{i=1}^{m} (ECR_a * T_{a,i} + ECR_d * T_{d,i}) \tag{8}$$

The fifth performance metric is the average difference in remaining energy between two servers to evaluate how balanced the system is in terms of energy consumption. The more balanced the system is, the longer lifetime the system has. This metric is computed using the following formula, where m is the total number of servers, and RE_i and RE_j are the remaining energy of servers S_i and S_j, respectively.

$$Average\ difference\ in\ remaining\ energy\ between\ two\ servers\ = \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} |RE_i - RE_j|}{(m-1)*m} \tag{9}$$

5.2 Simulation Results

Fig. 3 shows in both SODA and SESAMO, that the total time when servers are in active mode increases when the transaction inter-arrival time increases. The total time of SESAMO is always much longer than SODA's because SESAMO is pessimistic and uses locks to hold limited system resources to prevent conflicting transactions from accessing them. In other words, servers have to be in active mode most of time to keep processing transactions.

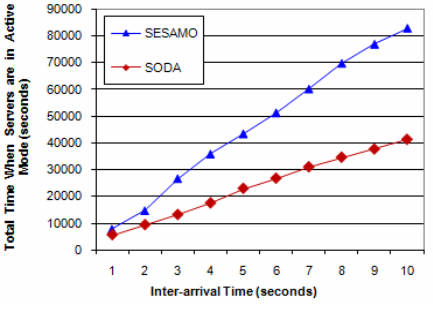


Fig. 3. The total time when servers are in active mode vs. inter-arrival time

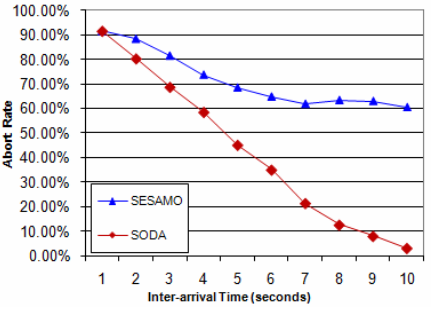


Fig. 4. The abort rate vs. inter-arrival time

Fig. 4 shows that the abort rates of SODA and SESAMO decrease when the transaction inter-arrival time increases. The abort rate of SODA is much lower than that of SESAMO right after the inter-arrival time is longer than 1 second. This is mainly because SODA is optimistic and non-blocking, and conflicts among transactions become rare, so that servers are not in active mode most of time (as shown in Fig. 3) and can process transactions in time. Although SESAMO does not enforce global serializability, strict 2PL running both locally and globally still blocks many conflicting transactions. When the inter-arrival time is getting shorter, it is easy to see that the abort rate of SODA is close to SESAMO's because conflicts among transactions increase; in addition, this confirms the fact that optimistic CC techniques work well only if conflicts among transactions are rare.

Fig. 5 shows the total number of cluster head reelections of SODA increases as the inter-arrival time increases. When the inter-arrival time reaches 10 seconds, the total simulation time is close to 3 hours (1000 transactions * 10 seconds = 10,000 seconds). Consequently, more cluster heads' remaining energy is below the predefined threshold *LET*, and more reelections are triggered to change roles for preserving energy. However, the total number of reelections of SESAMO is always zero because its design does not involve any cluster heads. In other words, SESAMO does not rotate roles among servers to balance energy.

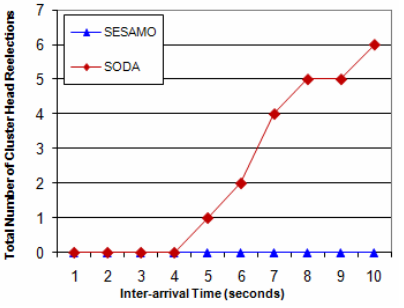


Fig. 5. The total number of cluster head reelections vs. inter-arrival time

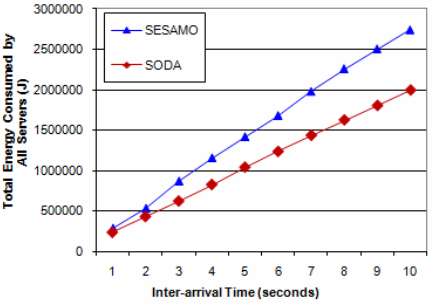


Fig. 6. The total energy consumed by all servers vs. inter-arrival time

Fig. 6 shows that the total energy consumption of all servers increases with the increase of the inter-arrival time. This is expected because more transactions are committed as inter-arrival time increases as shown in Fig. 4, so that each server has to spend more time in active mode on processing these committed transactions as shown in Fig. 3. SODA consumes at least 51,124 J and at most 749,727 J less than SESAMO right after when the inter-arrival time is longer than 1 second. This happens because transactions arrive into the system with a slow rate, and conflicts among transactions become much rarer, so that optimistic SODA performs better than pessimistic SESAMO due to no prevention of conflicts overhead.

The average difference in energy consumption between two servers when varying the inter-arrival time is shown in Fig. 7. Through this metric, we want to check whether the energy consumption is balanced among servers.

It is easy to see that SODA balances remaining energy better than SESAMO. This is because more non-primary cluster heads and primary cluster heads with higher energy are reelected as shown in Fig. 5. However, in SESAMO, there is no role rotation strategy and clients may keep submitting transactions to the same servers so that these servers are overloaded.

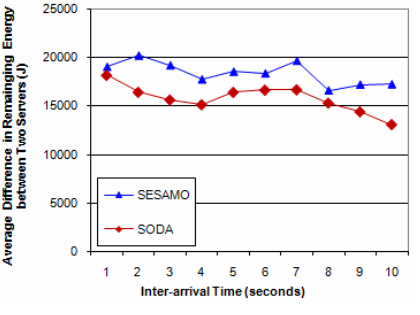


Fig. 7. The average difference in remaining energy between two servers vs. inter-arrival time

6 Conclusion and Future Research

In this paper, we introduced a database transaction concurrency control technique, called SODA, that can be used to support mission-critical applications such as disaster rescue and battlefields in a clustered MANET. This technique considers transaction real-time constraints as well as mobility, energy limitation, and workload of both

mobile servers and mobile clients in a clustered MANET architecture. Our solution is aimed at reducing transaction abort rate while saving the energy consumption by servers and balancing the energy consumption among servers. With respect to these performance metrics, the simulation results show the superiority of SODA over the existing technique, SESAMO.

For future research, we plan to incorporate data replication into our model to improve data access time and availability. We will also investigate the impacts of mobility (speed) of mobile nodes, disconnection time and read-only transaction percentage on abort rate, response time, total energy consumed by all servers, and average difference in remaining energy between two servers. We also plan to investigate alternative commit protocols.

Acknowledgment. This material is based upon work supported by (while serving at) the National Science Foundation (NSF) and the NSF Grant No. IIS-0312746.

References

1. Alampalayam, S.P., Srinivasan, S.: Intrusion Recovery Framework for Tactical Mobile Ad hoc Networks. The International Journal of Computer Science and Network Security 9(9), 1–10 (2009)
2. Catarci, T., De Leoni, M., Marrella, A., Mecella, M., Salvatore, B., Vetere, G., Dustdar, S., Juszczyk, L., Manzoor, A., Truong, H.: Pervasive Software Environments for Supporting Disaster Responses. IEEE Internet Computing 12(1), 26–37 (2008)
3. Lu, W., Seah, W.K.G., Peh, E.W.C., Ge, Y.: Communications Support for Disaster Recovery Operations using Hybrid Mobile Ad-Hoc Networks. In: Proceedings of the 32nd IEEE Conference on Local Computer Networks, Dublin, Ireland, pp. 763–770 (2007)
4. Silberschatz, A., Korth, H.F., Sudarshan, S.: Database Systems Concepts. McGraw-Hill College, New York (2005)
5. Chlamtac, I., Conti, M., Liu, J.: Mobile Ad Hoc Networking: Imperatives and challenges. Ad Hoc Networks Publication 1(1), 13–64 (2003)
6. Sklavos, N., Touliou, K.: Power Consumption in Wireless Networks: Techniques & Optimizations. In: Proceedings of The IEEE Region 8, International Conference on "Computer as a Tool", EUROCON 2007 (2007)
7. Choi, M., Park, W., Kim, Y.: Two-phase Mobile Transaction Validation in Wireless Broadcast Environments. In: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, pp. 32–38 (2009)
8. Lam, K., Wong, C.S., Leung, W.: Using Look-ahead Protocol for Mobile Data Broadcast. In: Proceedings of the 3rd International Conference on Information Technology and Applications, pp. 342–345 (2005)
9. Lei, X., Zhao, Y., Chen, S., Yuan, X.: Scheduling Real-Time Nested Transactions in Mobile Broadcast Environments. In: Proceedings of the 9th International Conference for Young Computer Scientists, pp. 1053–1058 (2008)
10. Brayner, A., Alencar, F.S.: A Semantic-serializability Based Fully-Distributed Concurrency Control Mechanism for Mobile Multi-database Systems. In: Proceedings of the 16th International Workshop on DEXA, pp. 1085–1089 (2005)
11. Xing, Z., Gruenwald, L., Phang, K.K.: A Robust Clustering Algorithm for Mobile Ad-hoc Networks. In: Pierre, S. (ed.) Handbook of Research on Next Generation Mobile Networks and Ubiquitous Computing, pp. 187–200. IGI Global (2010) ISBN: 160566250X

12. Wireless Ad Hoc Technology, http://www.atacwireless.com/adhoc.html
13. Chatterjee, M., Das, S.K., Turgut, D.: WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. Cluster Computing 5(2), 193–204 (2002)
14. Basu, P., Khan, N., Little, T.D.C.: A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks. In: Proceedings of IEEE ICDC, pp. 413–418 (2001)
15. Xue, M., Er, I., Seah, W. K. G.: Analysis of Clustering and Routing Overhead for Clustered Mobile Ad Hoc Networks. In: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, pp. 46–53 (2006)
16. Wang, Y., Kim, M.S.: Bandwidth-adaptive Clustering for Mobile Ad Hoc Networks. In: Proceedings of International Conference on Computer Communications and Networks, pp. 103–108 (2007)
17. Xing, Z., Gruenwald, L., Phang, K.K.: SODA: an Algorithm to Guarantee Correctness of Concurrent Transaction Execution in Mobile P2P Databases. In: Proceedings of the 19th International Conference on DEXA Workshop, pp. 337–341 (2008)
18. Hwang, S.: On Optimistic Methods for Mobile Transactions. Journal of Information Science and Engineering 16(4), 535–554 (2000)
19. Holanda, M., Brayner, A., Fialho, S.: Introducing self-adaptability into transaction processing. In: Proceedings of the 2008 ACM Symposium on Applied computing, pp. 992–997 (2008)
20. Pritsker, A., O'Reilly, J.: Simulation with Visual SLAM and AweSim, 2nd edn. John Wiley & Sons, New York (1999)
21. Gruenwald, L., Banik, S.M., Lau, C.N.: Managing real-time database transactions in mobile ad-hoc networks. Distributed and Parallel Databases Journal 22(1), 27–54 (2007)
22. Leu, Y., Hung, J.J., Lin, M.B.: A New Cache Invalidation and Searching Policy for mobile Ad Hoc Networks. In: Proceedings of the 2007 Annual Conference on International Conference on Computer Engineering and Applications, pp. 337–343 (2007)
23. Notebookcheck, http://www.notebookcheck.net/Review-Lenovo-ThinkPad-T400s-Notebook.21081.0.html

An Ontology-Based Method for Duplicate Detection in Web Data Tables

Patrice Buche[1], Juliette Dibie-Barthélemy[2],
Rania Khefifi[3], and Fatiha Saïs[3]

[1] INRA - UMR IATE, 2, place Pierre Viala, F-34060 Montpellier Cedex 2, France
LIRMM, CNRS-UM2, F-34392 Montpellier, France
Patrice.Buche@supagro.inra.fr
[2] INRA - Mét@risk & AgroParisTech,
16 rue Claude Bernard, F-75231 Paris Cedex 5, France
Juliette.Dibie@agroparistech.fr
[3] LRI (CNRS & Paris-Sud 11 University)/INRIA Saclay,
4 rue Jacques Monod, bât. G, F-91893 Orsay Cedex, France
{Rania.Khefifi,Fatiha.Sais}@lri.fr

Abstract. We present, in this paper, a duplicate detection method in semantically annotated Web data tables, driven by a domain Termino-Ontological Resource (TOR). Our method relies on the fuzzy semantic annotations automatically associated with the Web data tables. A fuzzy semantic annotation is automatically associated with each row of a Web data table. It corresponds to the instantiation of a composed concept of the domain TOR, which represents the semantic n-ary relationship that exists between the columns of the Web data table. A fuzzy semantic annotation contains fuzzy values expressed as fuzzy sets. We propose an automatic duplicate detection method which consists in detecting the pairs of duplicate fuzzy semantic annotations and relies on (i) knowledge declared in the domain TOR and on (ii) similarity measures between fuzzy sets. Two new similarity measures are defined to compare both, the symbolic fuzzy values and the numerical fuzzy values. Our method has been tested on a real application in the domain of chemical risk in food.

1 Introduction

Today's Web is not only a set of semi-structured documents interconnected via hyper-links. A huge amount of scientific and technical documents, available on the Web or on the hidden Web (digital libraries, ...), include structured data represented in data tables. Those data tables can be seen as small relational databases even if they lack the explicit metadata associated with a database. They represent a very interesting potential external source for building a data warehouse dedicated to a given application domain. They can be used to enrich local data sources or to compare local data with external ones. In order to integrate data, a preliminary step consists in harmonizing the vocabulary of the external data with the vocabulary of the local data, which is represented by a

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 511–525, 2011.

domain ontology. Therefore, external and local data can be indexed and queried using the same vocabulary. In [1], Hignette and al. have developed an automatic and ontology-based method for semantic annotation of Web data tables. The obtained annotations are expressed thanks to the domain ontology and fuzzy (see [2], for more details on fuzzy sets). Fuzzy annotations may have two different semantics: they represent either data imprecision or similarities between terms of data tables and terms of the ontology.

The semantic annotation allows the integration of Web external data with local ones, solving the vocabulary heterogeneity problem, but it does not prevent the integration of duplicate data into the data warehouse. The presence of duplicates in the data warehouse impacts the data quality and therefore the results of their exploitation (for instance, data analysis and decision aid). We propose in this paper to study the duplicate detection problem in Web data tables, using the fuzzy semantic annotations associated with the data tables thanks to a domain ontology. We propose an automatic method of duplicate detection which relies on (i) knowledge declared in the domain ontology, as it is done in [3], and on (ii) similarity measures between fuzzy sets.

The result of this work has been integrated in the @Web system which was previously developed (see [4,1,5]). @Web system is based on the semantic Web framework[1] and language recommendations (XML, RDF, OWL), which allow an XML/RDF data warehouse to be supplemented with Web data tables, as presented in Figure 1. @Web system relies on a domain Termino-Ontological Resource (TOR) manually built by domain experts.

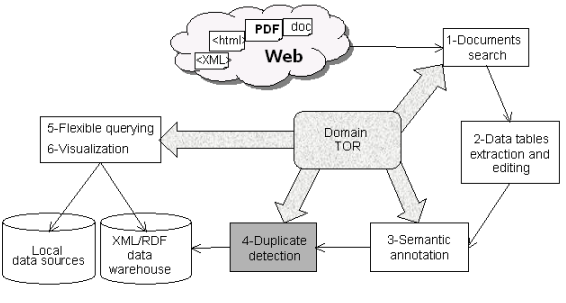


Fig. 1. Main steps of the ONDINE system

We will present in this paper how @Web system can be extended with a new duplicate detection step using the fuzzy semantic annotations associated with the Web data tables. We suppose that the Web data tables were previously automatically annotated thanks to the annotation method described in [1]. In section 2, we briefly present the domain TOR, we recall the semantic annotation method of @Web system (see [1]), and we recall the reference reconciliation

[1] http://www.w3.org/standards/semanticweb/

method (N2R) of [3] on which relies our work. In section 3, we present our duplicate detection method in Web data tables driven by a domain TOR. In section 4, we present some experiment results obtained on real data of chemical risk in food domain. We conclude and present some future work in section 5.

2 Preliminaries

In subsection 2.1, we present the domain TOR. In subsection 2.2, we recall the semantic annotation method of the @Web system presented in [1]. Finally, in subsection 2.3, we recall the numerical reference reconciliation method N2R [3].

2.1 The Domain Termino-Ontological Resource

A Termino-Ontological Resource (TOR) [6,7] is composed of a conceptual component and a terminological component. The conceptual component represents the ontology of the TOR. It is composed of two main parts: a generic part, commonly called *core ontology*, which contains the structuring concepts of the data table integration task, and a specific part, commonly called *domain ontology*, which contains the concepts that are specific to the domain of interest. The core ontology is composed of three kinds of *generic concepts*:

1. *simple concepts* which contain the symbolic concepts and the numerical concepts. Symbolic concepts are hierarchically organized by the "is-a" relationship. A numerical concept is described by a set of units, which are sub concepts of the *unit concept*, and eventually a numerical interval;
2. *unit concepts* which contain the units used to characterize the numerical concepts;
3. *composed concepts* which allow n-ary relationships to be represented between simple concepts. A composed concept is described by a signature, which is defined by a domain and a range. The domain contains one or several simple concepts, called *access concepts*, while the range contains only one simple concept, called *result concept*. A composed concept is denoted by $CC(Aa_1, Aa_2, \ldots, Aa_n, Ar)$ where CC is the name of the composed concept and $(Aa_1, Aa_2, \ldots, Aa_n, Ar)$ represents its signature: $Aa_1, Aa_2, \ldots, Aa_n$ are the access concepts of CC and Ar its result concept. The simple concepts which belong to the signature of a composed concept can be declared as important or simply optional using FOL Horn rules.

The concepts belonging to the domain ontology, called *specific concepts*, appear in the domain TOR as sub concepts of the generic concepts.

In the domain TOR, all concepts are represented by OWL classes. The Horn rules are expressed using SWRL rules (Semantic Web Rule Language) recommended by the W3C[2]. The disjunction constraints, which can be declared between simple concepts and/or composed concepts, are expressed using OWL

[2] http://www.w3.org/Submission/SWRL/

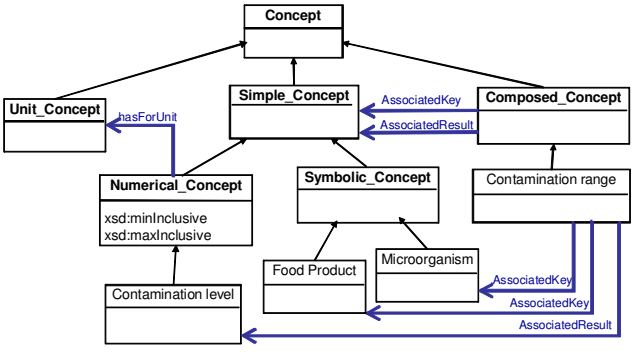


Fig. 2. Conceptual component of a TOR in the domain of chemical risk in food

constructor *owl:disjointWith*. Figure 2 gives an example of the conceptual component of a TOR in the domain of chemical risk in food. The concepts belonging to the core ontology are represented in bold.

The terminological component is the terminology of the TOR: it contains the term set of the domain of interest. A term is defined as a sequence of words, in a language, and has a label.

2.2 Semantic Annotation of Web Data Tables Driven by a Domain TOR

A data table is composed of columns, themselves composed of cells. The cells of a data table may contain terms or numerical values often followed by a measure unit. The semantic annotation of a Web data table consists in annotating cells content, in order to identify the symbolic or numerical concepts represented by its columns and finally the semantic n-ary relationships between its columns.

Table 1. Example of a Web data table

Food	*Contaminant*	*Max Value (µg/kg)*	*Contamination Level (µg/kg)*
Breakfast cereals	Ochratoxin A	6	<0.2
Baby food	Patulin	58	6.3

Example 1. *Table 1 presents an example of a Web data table in which the composed concept* Contamination Range *was identified. The first line of the Web data table indicates that* Breakfast cereals *is contaminated by the* Ochratoxin A *at a* contamination level *smaller than 0.2 µg/kg.*

Several composed concepts of a domain TOR can be recognized to annotate a Web data table. The semantic annotation of a Web data table consists in instantiating each recognized composed concept for each row of the table. A composed

concept instantiation associated with a row of a Web data table include values expressed as fuzzy sets [2]. In a fuzzy set $\mathcal{A}$ defined on a domain $\mathcal{X}$, each element $x \in \mathcal{X}$ can belong partially to the fuzzy set with a membership degree, denoted $\mu_{\mathcal{A}}(x)$, between 0 (element which is not part of the fuzzy set) and 1 (element which is completely part of the fuzzy set). The definition domain $\mathcal{X}$ can be continuous or discrete. The support $S(\mathcal{A})$ and the kernel $K(\mathcal{A})$ of the fuzzy set $\mathcal{A}$ are the sets: $S(A) = \{x \in \mathcal{A} | \mu_{\mathcal{A}}(x) > 0\}$ and $K(\mathcal{A}) = \{x \in \mathcal{A} | \mu_{\mathcal{A}}(x) = 1\}$. The fuzzy values, found in the composed concept instantiations, may express two of the three classical semantics of fuzzy sets [8]: similarity or imprecision. A discrete fuzzy set with a semantics of similarity is associated with each cell belonging to a column recognized as symbolic. It represents the ordered list of the most similar terms of the domain TOR associated with the original term present in the cell. A continuous fuzzy set with a semantics of imprecision may be associated with cells belonging to columns recognized as numerical ones. It represents an ordered disjunction of exclusive possible values.

Definition 1. A **discrete fuzzy set** $\mathcal{A}$, denoted DFS, is a fuzzy set associated with a symbolic concept of the domain TOR. Its definition domain is the set of terms of the domain TOR. We denote by $\{x_1/y_1, \ldots, x_n/y_n\}$ the fact that element x_k has membership degree y_k.

Definition 2. A **continuous fuzzy set** $\mathcal{A}$, denoted CFS, is a trapezoidal fuzzy set associated with a numerical concept of the domain TOR. A trapezoidal fuzzy set $\mathcal{A}$ is defined by its four (ordered) characteristic points $[a, b, c, d]$ which correspond to its support $[a, d]$ and its kernel $[b, c]$ (see Figure 3). Its definition domain is the interval of possible values for the numerical concept.

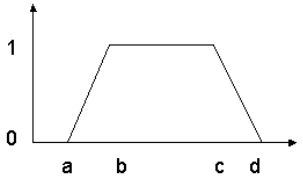


Fig. 3. A trapezoidal continuous fuzzy set

Example 2. *The discrete fuzzy set associated with the term* "Breakfast cereal" *of the first row of Table 1 is: { breakfast cereal sweet/0.602, breakfast cake/0.5, cereal bar chocolat/0.408, cereal bar/0.5, cereal bar low calorie/0.354 }. The continuous fuzzy set associated with the numerical value* "< 0.2" *of the first row of Table 1 is: [0, 0, 0.2, 0.2].*

2.3 Reference Reconciliation Method (N2R)

To develop a duplicate detection method we have chosen to rely on reference reconciliation methods which are automatic and ontology based, in order to

benefit from the knowledge which is declared in the domain TOR. N2R method, developed by Saïs and al [3], is a method which has two main distinguishing characteristics. First of all, it is fully unsupervised, i.e., it does not require any training phase from manually labeled data to set up coefficients or parameters. Secondly, two functions modeling the influence between similarities of references take into account the constraints associated with the functional properties declared in the OWL ontology in a declarative way. Furthermore, ontology and data knowledge (disjunctions and Unique Name Assumption) are exploited by N2R in a filtering step to reduce the number of reference pairs which are considered in the similarity computation step.

The duplicate detection method, present in the following, relies on N2R method in the sense that it exploits knowledge declared in the domain TOR to both, (i) filter the pairs of data to be compared, thanks to disjunctions declared in the domain TOR, and (ii) to express the influence degrees existing between the different similarities, thanks to the declaration of concept importance.

3 Duplicate Detection Method

We present in this section our duplicate detection method. Our method takes as input two Web data tables which were previously automatically semantically annotated thanks to a domain TOR using the method of [1]. Since each data table is annotated by a set of composed concept instances, our method consists in detecting the pairs of duplicate composed concept instances by comparing them two by two. We first present the definitions of simple concept instances and composed concept instances. Since the composed concept instances contain fuzzy values, we then propose two new similarity measures to compare, on the one hand, the discrete fuzzy sets and, on the other hand, the continuous fuzzy sets. We finally present the algorithms of our method and an illustrative example.

3.1 Definitions of Simple and Composed Concept Instances

The input of our method is a set of composed concept instances associated with each Web data table to be compared. A composed concept instance is composed of the instances of the simple concepts which belong to its signature.

Definition 3. *A* simple concept instance*, denoted* $inst_{c_i}$ *where* c_i *is a simple concept* ($c_i = SimpleConcept(inst_{c_i})$)*, can be represented by either:*

- *a discrete fuzzy set having a semantics of similarity which is composed of a set of terms* t_k *of the domain TOR with their membership degrees* d_k*:* $inst_{c_i} = (c_i, \{ t_1/d_1, \ldots, t_n/d_n \})$*;*
- *or a continuous trapezoid fuzzy set having a semantics of imprecision which is described by its support* $[sup_{min}, sup_{max}]$ *and its kernel* $[ker_{min}, ker_{max}]$*:* $inst_{c_i} = (c_i, [sup_{min}, ker_{min}, ker_{max}, sup_{max}])$.

We can therefore give the definition of a composed concept instance.

Definition 4. *A* composed concept instance*, denoted ICC, is a couple (id, $descr_{id}$) where id is the ID associated with the composed concept CC and $descr_{id}$ its description. The description of a composed concept is the set of the instances of the simple concepts which belong to its signature: $descr_{id}$ = { (c_1, $inst_{c_1}$), ..., (c_n, $inst_{c_n}$) }. We denoted id.inst the set of the simple concept instances: $id.inst = \{inst_{c_1}, \ldots, inst_{c_n}\}$*

Example 3. *The description of the composed concept instance associated with the first row of Table 1 is: { (Food Product, { breakfast cereal sweet/0.602, breakfast cake/0.5, cereal bar chocolat/0.408, cereal bar/0.5, cereal bar low calorie/0.354 }), (Contaminant, { Ochratoxin A/1 }), (Contamination level, [0, 0, 0.2, 0.2]) }.*

3.2 Two Similarity Measures to Compare Fuzzy Sets

In this section, we propose two similarity measures to compare, on the one hand, discrete fuzzy sets, and on the other hand, continuous fuzzy sets. [9] proposed a classification of comparison measures between fuzzy objects into four categories: satisfiability, inclusion, resemblance and dissimilarity. In this paper, we are looking for a measure of resemblance, which is a measure of similitude between two fuzzy sets looking at the characteristics they have in common, without regarding one of them as a reference. In [9], this family of measures satisfies the two properties of reflexivity and symmetry, which can be easily checked for the two measures we propose in the following.

A similarity measure to compare discrete fuzzy sets. There exist several similarity measures between sets of terms (see [10]). We can cite, in particular, the measure of Jaccard [11], the measure of Tversky [12] or the measure of Soft-Jaccard [13], which allow the comparison between sets of terms. The measure we have to choose must take into account the fact that the discrete fuzzy sets we want to compare are sets of terms associated with membership degrees (see definition 3). We therefore propose a new similarity measure, called *Sim*, which is inspired from the Jaccard measure. The Jaccard measure is defined as the intersection (number of common terms) divided by the union (total number of terms) of the two sets to compare. In our *Sim* measure, the number of common terms corresponds to the sum of the minimum degrees associated with the common terms of both fuzzy sets. The total number of terms corresponds to the sum of the maximum degrees associated with the terms of the fuzzy sets. These are the classical ways to represent the intersection and the union of two fuzzy sets. Let A and B be two discrete fuzzy sets, $deg_A(t)$ (respectively $deg_B(t)$) the membership degree of the term t to the fuzzy set A (respectively B), the similarity measure *Sim* is defined as follows:

$$Sim(A,B) = \frac{\sum_{t \in A \cap B} \min(deg_A(t), deg_B(t))}{\sum_{t \in A \cup B} \max(deg_A(t), deg_B(t))} \quad (1)$$

A similarity measure to compare continuous fuzzy sets. There exist several similarity measures between continuous fuzzy sets. We can cite, in particular, the measure of Hsieh and Chen [14], the measure of Chen [15] and the measure of Chen and Chen [16]. The measure we have to choose must take into account two constraints on the continuous fuzzy sets we want to compare. The first constraint to be considered is that continuous fuzzy sets are not necessarily normalized, i.e. their values are not necessarily included between 0 and 1. We can cite for instance the numerical concept pH whose values belong to [0, 14]. In the second constraint, redundancies between continuous fuzzy sets must be detected even if they represent values with a different precision scale. For instance, a table may contain the mean value of repeated experimental data whereas, in a redundant table, the value is expressed by a mean value and associated standard deviation. Since the measure of Chen [15] does not allow the comparison between not normalized continuous fuzzy sets and the measure of Hsieh and Chen [14] does not allow the comparison between continuous fuzzy sets of different precision scales, we propose to use the measure of simple center of gravity method (SCGM) of Chen and Chen [16]. This measure relies on a similarity measure between the center-of-gravity points of the fuzzy sets to compare. Let (a_1, a_2, a_3, a_4) be a continuous trapezoid fuzzy set, the coordinates x^* and y^* of the center-of-gravity points are computed by the SCGM method as follows:

$$If\ a_1 = a_4 \longrightarrow \begin{cases} y^* = 1/2 \\ x^* = a_1 \end{cases}, otherwise \longrightarrow \begin{cases} y^* = \frac{\frac{a_3 - a_2}{a_4 - a_1} + 2}{6} \\ x^* = \frac{y^*(a_3 + a_2) + (a_4 + a_1)(1 - y^*)}{2} \end{cases} \quad (2)$$

In order to compute the similarity measure between two continuous trapezoid fuzzy sets A and B, denoted $Sim(A, B)$, we propose to use the distance between their center-of-gravity points as follows:

$$\begin{aligned} &Sim(A, B) = \frac{1}{1 + d(cent_A, cent_B)} \\ &where\ d(cent_A, cent_B) = \sqrt{(x_A^* - x_B^*)^2 + (y_A^* - y_B^*)^2} \end{aligned} \quad (3)$$

3.3 The Duplicate Detection Algorithm

We now detail our duplicate detection method between two Web data tables which were semantically annotated thanks to a domain TOR. Our method consists in detecting the pairs of duplicate composed concepts instances, which are associated with the Web data tables. To do that, we propose to compute a similarity score between the descriptions of each pair of composed concept instances. This similarity score relies on the similarity measures presented in subsection 3.2 and on knowledge declared in the domain TOR. Algorithm 1 presents the main steps of our duplicate detection method.

Algorithm 1 requires three kinds of inputs. Let T_1 and T_2 be two Web data tables semantically annotated thanks to a domain TOR. The first input is the two sets of composed concept instances $Set_1(ICC)$ and $Set_2(ICC)$ which are

Algorithm 1. Duplicate detection Algorithm

Input: – $Set_1(ICC)$: set of composed concept instances associated with the first Web data table T_1
– $Set_2(ICC)$: set of composed concept instances associated with the second Web data table T_2
– $Disj$: set of disjunction constraints between the concepts of the domain TOR
– Tax : hierarchical relationships between simple concepts in the domain TOR
– $ImportantSimpleConcepts$: set of the signatures of the composed concepts in the domain TOR with their important simple concepts
– T_{dup}: predefined threshold of the duplicate decision
Output: – set of duplicate pairs of composed concept instances
{1: *building of the set of pairs of comparable composed concept instances*}
$S \leftarrow comparableICCPairs(Set_1(ICC), Set_2(ICC), Disj)$
$DUP \leftarrow \emptyset$
{*2: computation of the similarity score*}
For Each $(icc_1, icc_2) \in S$ ***Do***
$score \leftarrow SimilarityScore((icc_1, icc_2), Disj, Tax, ImportantSimpleConcepts)$
{*3: duplicate decision*}
If $score > T_{dup}$ ***Then***
$DUP \leftarrow DUP \cup (icc_1, icc_2)$
EndIf
End Each
return DUP

respectively associated with the Web data tables T_1 and T_2. The second kind of input corresponds to the knowledge declared in the domain TOR: (1) the disjunctions between composed concepts and the disjunctions between simple concepts, which allows one to avoid some obvious comparisons between composed concept instances and between simple concept instances, (2) the hierarchical relationships between simple concepts represented by a taxonomy, (3) the importance of the simple concepts in the signatures of the composed concepts. The third kind of input is a predefined threshold used to determine if two composed concept instances are duplicate or not according to their similarity score. Algorithm 1 has for output the set of duplicate pairs of composed concept instances. The first step of Algorithm 1 consists in building the set of pairs of comparable composed concepts instances using the disjunction constraints defined in the domain TOR. Two composed concept instances icc_1 and icc_2 are said *comparable* if the composed concepts cc_1 and cc_2 are not declared as disjoints in the TOR. A similarity score is then computed for each pair of comparable composed concept instances (step 2). The computation of this score is detailed in Algorithm 2 presented below. Finally, two composed concept instances are said redundant if the similarity score between their descriptions is greater than a given threshold (step 3).

Algorithm 2 gives details on the similarity score computation for one pair of comparable composed concept instances. This score is computed thanks to the similarity measures, presented in subsection 3.2, between each pair of comparable simple concept instances, which belong to the signatures of the composed concept

Algorithm 2. Computation of the similarity score for a pair of comparable composed concept instances

Input: – (icc_1, icc_2) : pair of composed concept instances
– $Disj$: set of disjunction constraints between the concepts of the domain TOR
– Tax : hierarchical relationships between simple concepts in the domain TOR
– $ImportantSimpleConcepts$: set of the signatures of the composed concepts in the domain TOR with their important simple concepts

Output: similarity score of the pair of comparable composed concept instances (icc_1, icc_2)

$f_{imp} \leftarrow 1$
$Score_{Nimp} \leftarrow 0$
{*1: Computation of the similarity scores between each pair of comparable simple concept instances*}
For Each $a \in id_1.inst$ ***Do***
  $best \leftarrow \emptyset$
  For Each $b \in id_2.inst$ ***Do***
    $score_{max}(a, b) \leftarrow 0$
    If *ComparableSimpleConcepts(a, b, Disj)* ***Then***
      $score_{Inst}(a, b) = Sim(a, b)$
      If $SimpleConcept(a) \neq SimpleConcept(b)$ ***Then***
        $score_{Sem}(a, b) = \frac{1}{1+LCS(SimpleConcept(a), SimpleConcept(b), Tax)}$
        $score_{final}(a, b) = \frac{score_{Sem}(a,b)+score_{Inst}(a,b)}{2}$
      Else
        $score_{final}(a, b) = score_{Inst}(a, b)$
      EndIf
      If $score_{final}(a, b) > score_{max}(a, b)$ ***Then***
        $best \leftarrow b$
        $score_{max}(a, best) \leftarrow score_{final}(a, b)$
      EndIf
    EndIf
  End Each
  {*2: Computation of the similarity score of the pair of comparable composed concept instances* (icc_1, icc_2)}
  If *(Is_Important(a, ImportantSimpleConcepts) and Is_Important(best, ImportantSimpleConcepts))* ***Then***
    $f_{imp} = f_{imp} \times score_{max}(a, best)$
  Else
    $Score_{Nimp} = Score_{Nimp} + score_{max}(a, best)$
  EndIf
End Each
$f_{Nimp} = \frac{Score_{Nimp}}{\max(|id_1.inst|,|id_2.inst|)}$
$S = \max(f_{imp}, f_{Nimp}))$
return S

instances. Two simple concept instances are said *comparable* if the corresponding simple concepts are not declared as disjoint in the domain TOR. In the first step of Algorithm 2, a similarity score is computed for each simple concept instance a, which belongs to the composed concept instance icc_1 ($a \in id_1.inst$), with each simple concept instance b, which belongs to the composed concept instance icc_2 ($b \in id_2.inst$). This score is a combination of:

1. a semantic similarity score $score_{sem}$ between the simple concepts associated with a and b, which relies on the notion of lowest common subsumer (LCS) in the hierarchy of simple concepts in the domain TOR;
2. an instance score $score_{inst}$ which is computed thanks to the similarity measures $Sim(a, b)$, presented in subsection 3.2, depending upon the simple concepts associated with a and b are symbolic or numerical.

For each simple concept instance $a \in id_1.inst$, we keep the best similarity score with the simple concept instances $b \in id_2.inst$. We can therefore compute the similarity score of the pair of comparable composed concept instances (icc_1, icc_2) (step 2). This similarity score is computed thanks to the importance of the simple concepts in the signatures of the composed concepts associated with icc_1 and icc_2, defined in the domain TOR. It is a combination of (i) a similarity score f_{imp} for the instances of the simple concepts which are declared as important, computed as the product of the similarity scores of their pairs of instances and (ii) a similarity score f_{Nimp} for the instances of the simple concepts which are not declared as important, computed as the average value of the similarity scores of their pairs of instances.

3.4 An Illustrative Example of Our Duplicate Detection Method

To illustrate our method, let us consider Table 1 presented in subsection 2.2 and Table 2 presented below.

Table 2. Example of a Web data table (T2)

Food	*Contaminant*	*Year*	*Lod*	*Contamination Level*
Baby food	Patulin	2000	0.7	6.3
Apple juice	Patulin	1998	2	8.37
Breafast cereal	Ochratoxin A	2003	0.7	<0.2

The identified composed concepts in Table 1 are the following:

- ContaminationRange (**Food**, **Contaminant**, year, **ContaminationLevel**)
- LodRelation (**Food**, **Contaminant**, year, **SamplesTotalNumber**, **lod**)

The identified composed concepts in Table 2 are the following:

- ContaminationRange (**Food**, **Contaminant**, year,**ContaminationLevel**)

– MaxContaminationRange (**Food**, **Contaminant**, year, **MaxContaminationlevel**)

The simple concepts in bold represent the important simple concepts of the signature of the composed concepts. We suppose that (i) the composed concepts are declared as pairwise disjoint in the TOR except the composed concepts *ContaminationRange* and *MaxContaminationRange* and (ii) the simple concepts are declared as pairwise disjoint in the TOR except the simple concepts *ContaminationLevel* and *MaxContaminationLevel*. In the following, the composed concept instances and the simple concept instances are denoted by the number of their table and the number of their row. For instance, ICC_{n_T,n_L} corresponds to the instance of the composed concept CC in Row n_L of Table n_T.

We first identify the pairs of comparable composed concept instances according to the disjunction constraints defined in the domain TOR. For simplicity reason, we only consider in the following the pair ($ContaminationRange_{1,3}$, $ContaminationRange_{2,1}$). The descriptions associated with the two composed concept instances are:
$descr_{1,3}$={ ($FoodProduct_{1,3}$, { "breakfast cereal sweet"/0.408, "cereal bar chocolat"/0.408, "cereal bar"/0.5, "cereal bar low calorie"/0.354 }), ($Contaminant_{1,3}$, { "Ochratoxin" A/1 }), ($year_{1,3}$, { [2003, 2003] }), ($ContaminationLevel_{1,3}$, [0, 0, 0.2, 0.2]) }.
$descr_{2,1}$={ ($FoodProduct_{2,1}$, { "breakfast cereal sweet"/0.602, "breakfast cake"/0.5, "cereal bar chocolat"/0.408, "cereal bar"/0.5, "cereal bar low calorie"/0.354 }), ($Contaminant_{2,1}$, {"Ochratoxin A"/1 }), ($Contamination\ level_{2,1}$, [0, 0, 0.2, 0.2]) }.

We can now compute the similarity scores between each pair of comparable simple concept instances:
$score_{Inst}(FoodProduct_{1,3}, FoodProduct_{2,1}) = \frac{0.408+0.408+0.5+0.354}{0.602+0.5+0.408+0.5+0.354} = 0.7$
$score_{Inst}(Contaminant_{1,3}, Contaminant_{2,1}) = 1$
$score_{Inst}(ContaminationLevel_{1,3}, ContaminationLevel_{2,1}) = 1.$

Finally, we compute the similarity score of the pair ($ContaminationRange_{1,3}$, $ContaminationRange_{2,1}$) thanks to the importance of the simple concepts in the signature of the composed concept *ContaminationRange*:
$f_{Imp} = score_{Inst}(FoodProduct_{1,3}, FoodProduct_{2,1}) \times score_{Inst}(Contaminant_{1,3}, Contaminant_{2,1}) \times score_{Inst}(ContaminationLevel_{1,3}, ContaminationLevel_{2,1}) = 0.7 \times 1 \times 1 = 0.7$
$f_{NImp} = 0$. Then, we obtain $S = max(f_{Imp}, f_{NImp}) = 0.7$

If we set the duplicate threshold T_{dup} at 0.5, the third row of Table 2 and the first row of Table 1 are therefore duplicates with the similarity score of 0.7.

4 Experimentation

To evaluate the efficiency of our method we have applied the duplicate detection algorithm on several real Web Tables in the chemical risk in food domain. We will first give details on the dataset and then discuss the obtained results.

Table 3. Data set description

Tables	Identified composed concepts
T1	Lod, MaxContamination, MeanContaminationLevel, MedianContamination
T2	MaxContamination, MeanContaminationLevel, MedianContamination
T3	MaxContamination, MeanContaminationLevel, SamplesPositives, SdContaminationLevel
T4	MeanContaminationLevel, SamplesPositives, RangeContamination
T5	ContaminationLevel
T6	ContaminationLevel
T7	MeanContaminationLevel, SamplesPositives, RangeContamination

Dataset description. The considered data set is composed of seven Web tables that are annotated by @Web system by using the TOR of the chemical risk in food domain. In the Table 3 we give the web table list with the set of composed concepts that are identified within them.

The obtained results. We will present here the results obtained by applying our algorithm on the combinations of the above seven tables. We present here the results obtained by the following comparisons: (T1, T2), (T1, T3), (T4, T7) and (T5, T6). These combinations has been made in the way to combine tables having most of common composed concepts.

Table 4. Results in terms of recall, precision and F-Measure for the 4 combinations of tables: table (a) shows the best results for the 4 combinations and their corresponding threshold T_{dup} and table (b) shows the results for the 4 combinations where $T_{dup} = 0.7$

(a)

	Recall	Precision	F-measure	Best T_{dup}
(T1, T2)	1	1	1	1
(T4, T7)	1	1	1	1
(T5, T6)	1	1	1	0.75
(T1, T3)	-	-	-	-

(b)

	Recall	Precision	F-measure
(T1, T2)	1	1	1
(T4, T7)	1	0.59	0.74
(T5, T6)	1	0.54	0.7
(T1, T3)	-	-	-

We have computed the recall, the precision and the F-measure by comparing the results obtained by our method with the gold-standard results given by a domain expert. In Table 4 (a) we give the best results that are obtained for each pair of tables and their corresponding threshold T_{dup}. For the table pairs (T1, T2) and (T4, T7) we have obtained the maximum results, i.e. all the duplicate data have been detected by our method and all the detected duplicates are correct. These results are represented by a F-Meaure equals to 1 for a threshold equals to 1. For the table pair (T5, T6) we have obtained the maximum results where T_{dup} equals to 0.75. In Table 4 (b) we show the obtained results for the four combinations where T_{dup} is fixed at 0.7. We obtain the maximum results (F-measure equals to 1) for the tables T1 and T2. We obtain an F-meaure of 0.74 and 0.7 for the table pairs (T4, T7) and (T5, T6) respectively.

We note that comparisons between T1 and T3 correspond to the case of tables without duplicates. No duplicates have been detected by the method, which corresponds to the expected behaviour. It is denoted by a dash in Table 4 (a) and (b).

5 Conclusion

In this paper we have presented our automatic and ontology-based approach of duplicate detection in Web data tables. The originalities of this work is threefold: (i) the declarative way of exploiting ontology knowledge in the duplicate detection process, (ii) the development and the use of suitable similarity measures between numerical and symbolic fuzzy sets; and (iii) the ability to handle heterogeneous and imprecise data at different levels of granularity.

Our proposal in this paper can be compared to approaches studying the reference reconciliation problem, i.e., detecting whether different data descriptions refer to the same real world entity (e.g. the same person, the same paper, the same protein). Different approaches have been proposed. [17,18,10] have developed supervised reference reconciliation methods which use supervised learning algorithm in order to help the duplicate detection. Those methods require a set of reference pairs labeled as reconciled or not reconciled. [19,3] proposes a declarative approach which relies on expert knowledge expressed in an ontology and does need a learning phase. Since we have a domain TOR and we do not want to add a learning phase, we have proposed to extend the work of [3] in order to detect duplicates between data tables using their fuzzy semantic annotations. In a close domain to the references reconciliation, works have been done on data table fusion. [20,21], in particular, study the data integration into the Cloud in order to help end-users to collaboratively manage their data. Our approach is complementary since it detects duplicates between data tables which were extracted from the Web, before storing them in a data warehouse.

The efficiency of our duplicate detection method has been evaluated and validated on real data in the chemical risk in food domain. As future work, we plan to test our method on bigger data sets form different domains, in order to show its scalability and its generality. We aim also to study how information on data provenance (e.g., document authors, source reputation, etc) can help to improve the distinction between duplicate data, similar data and distinct data. Finally, it will be interesting to extend the proposed approach by studying how to deal with duplicate detection when data tables were annotated thanks to different ontologies.

References

1. Hignette, G., Buche, P., Dibie-Barthélemy, J., Haemmerlé, O.: Fuzzy annotation of web data tables driven by a domain ontology. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 638–653. Springer, Heidelberg (2009)

2. Zadeh, L.: Fuzzy sets. Information and Control 8, 338–353 (1965)
3. Saïs, F., Pernelle, N., Rousset, M.C.: Combining a logical and a numerical method for data reconciliation. J. Data Semantics 12, 66–94 (2009)
4. Buche, P., Haemmerlé, O.: Towards a unified querying system of both structured and semi-structured imprecise data using fuzzy view. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867, pp. 207–220. Springer, Heidelberg (2000)
5. Buche, P., Dibie-Barthélemy, J., Chebil, H.: Flexible sparql querying of web data tables driven by an ontology. In: Andreasen, T., Yager, R.R., Bulskov, H., Christiansen, H., Larsen, H.L. (eds.) FQAS 2009. LNCS, vol. 5822, pp. 345–357. Springer, Heidelberg (2009)
6. Roche, C., Calberg-Challot, M., Damas, L., Rouard, P.: Ontoterminology - a new paradigm for terminology. In: KEOD, pp. 321–326 (2009)
7. Reymonet, A., Thomas, J., Aussenac-Gilles, N.: Modelling ontological and terminological resources in OWL DL. In: OntoLex-Workshop at ISWC 2007 (2007)
8. Dubois, D., Prade, H.: The three semantics of fuzzy sets. Fuzzy Sets and Systems 90, 141–150 (1997)
9. Bouchon-Meunier, B., Rifqi, M., Bothorel, S.: Towards general measures of comparison of objects. Fuzzy Sets and Systems 11, 143–153 (1996)
10. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: KDD, pp. 39–48 (2003)
11. Jaccard, P.: Etude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin de la Société Vaudoise des Sciences Naturelles 37, 547–579 (1901)
12. Tversky, A.: Features of similarity. Psychological Review 84, 327–352 (1977)
13. Largeron, C., Kaddour, B., Fernandez, M.: Softjaccard: une mesure de similarité entre ensembles de chaînes de caractères pour l'unification d'entités nommées. In: Extaction et Gestion des Connaissances (EGC) (2009)
14. Hsieh, C.H., Chen, S.H.: Similarity of generalized fuzzy numbers with graded mean integration represntation. In: Proc. 8th IFSA World Congr., vol. 2, pp. 551–555 (1999)
15. Chen, S.M.: New methods for subjective mental workload assessment and fuzzy risk analysis. Cybernetics and Systems 27, 449–472 (1996)
16. Chen, S.J., Chen, S.M.: Fuzzy risk analysis based on similarity measures of generalized fuzzy numbers. IEEE 11(1), 45–56 (2003)
17. Cohn, D.A., Atlas, L.E., Ladner, R.E.: Improving generalization with active learning. Machine Learning 15(2), 201–221 (1994)
18. Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Inf. Syst. 26(8), 607–633 (2001)
19. Saïs, F., Pernelle, N., Rousset, M.C.: L2R: A logical method for reference reconciliation. In: AAAI Conference on Artificial Intelligence, pp. 329–334 (2007)
20. Gonzalez, H., Halevy, A.Y., Jensen, C.S., Langen, A., Madhavan, J., Shapley, R., Shen, W.: Google fusion tables: data management, integration and collaboration in the cloud. In: SoCC, pp. 175–180 (2010)
21. Gonzalez, H., Halevy, A.Y., Jensen, C.S., Langen, A., Madhavan, J., Shapley, R., Shen, W., Goldberg-Kidon, J.: Google fusion tables: web-centered data management and collaboration. In: SIGMOD Conference, pp. 1061–1066 (2010)

Approaches for Semantically Annotating and Discovering Scientific Observational Data*

Huiping Cao[1], Shawn Bowers[2], Mark P. Schildhauer[3]

[1] Dept. of Computer Science, New Mexico State University
hcao@cs.nmsu.edu
[2] Dept. of Computer Science, Gonzaga University
bowers@gonzaga.edu
[3] NCEAS, University of California Santa Barbara
schild@nceas.ucsb.edu

Abstract. Observational data plays a critical role in many scientific disciplines, and scientists are increasingly interested in performing broad-scale analyses by using data collected as part of many smaller scientific studies. However, while these data sets often contain similar types of information, they are typically represented using very different structures and with little semantic information about the data itself, which creates significant challenges for researchers who wish to discover existing data sets based on data semantics (observation and measurement types) and data content (the values of measurements within a data set). We present a formal framework to address these challenges that consists of a semantic observational model, a high-level semantic annotation language, and a declarative query language that allows researchers to express data-discovery queries over heterogeneous (annotated) data sets. To demonstrate the feasibility of our framework, we also present implementation approaches for efficiently answering discovery queries over semantically annotated data sets.

1 Introduction

Accessing and reusing observational data is essential for performing scientific analyses at broad geographic, temporal, and biological scales. Classic examples in earth and environmental science include examining the effects of nitrogen treatments across North American grasslands [17], and studying how changing environmental conditions affect bird migratory patterns [19]. These types of studies often require access to hundreds of data sets collected by independent research groups over many years. Tools that aim to help researchers discover and reuse these data sets must overcome a number of significant challenges: (1) observational data sets exhibit a high level of structural heterogeneity (e.g., see Fig. 1), which includes the use of various terms and conventions for naming columns containing similar or compatible information (e.g., "dw", "wt", "m", "biomass" may each be used to denote a "mass" measurement); and (2) semantic information about data sets, which is crucial for properly interpreting data, is typically either missing or only provided through natural-language descriptions.

* This work supported in part through NSF grants #0743429 and #0753144.

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 526–541, 2011.

site	*plt*	*size*	*ph*	*spp*	*len*	*dbh*
GCE6	A	7	4.5	piru	21.6	36.0
GCE6	B	8	4.8	piru	27.0	45
...	...	...	...	...	...	...
GCE7	A	7	3.7	piru	23.4	39.1
GCE7	B	8	3.9	piru	25.2	42.7
...	...	...	...	...	...	...

yr	*field*	*area*	*acidity*	*piru*	*abba*	...
2005	f1	5	5.1	20.8	14.1	...
2006	f1	5	5.2	21.1	15.2	...
...	...	...	...	...	...	...
2010	f1	5	5.8	22.0	18.9	...
2005	f2	7	4.9	18.9	15.3	...
...	...	...	...	...	...	...

Fig. 1. Typical examples of similar (but not identical) observational data sets consisting of study locations (plot, field), soil acidity measurements, and height and diameter measurements of trees

Despite these challenges, a number of efforts are being developed with the goal of creating and deploying specialized software infrastructures (e.g., [5,1]) to allow researchers to store and access observational data contributed from various disciplines. While a large number of data sets are stored using these repositories, these sites provide primarily simple keyword-based search interfaces, which for many queries are largely ineffective for discovering relevant data sets (in terms of precision and recall) [7].

This paper presents a formal semantic annotation and data discovery framework that can be used to uniformly represent and query heterogeneous observational data. We adopt an approach that is based on a number of emerging observation models (e.g., [3,11,9,15]), which provides canonical representations of observation and measurement structures that researchers can use to help describe, query, and access otherwise heterogeneous data sets. Here we consider the use of description-logic (i.e., OWL-DL) based ontologies for domain-specific terms to specify observation and measurement types that can be used to both annotate data sets and to specify data-discovery queries.

Semantic annotations in our framework define concrete mappings from relational data sets to a uniform observational model specialized by domain-specific terms. The annotation language was designed to support annotations created either manually or automatically (e.g., by employing attribute similarity measures or data-mining techniques). The language is currently being used to store annotations created (via a graphical user interface) within a widely used metadata editing tool [2] for earth and environmental science data sets. A key contribution of our annotation approach is that it provides a declarative, high-level language that follows the "natural" way in which users describe their observational data sets semantically (by focusing on attribute-level metadata, and inferring remaining structural relationships). We also support data-discovery queries posed over both the types of observations and measurements used to annotate data sets as well as over (possibly summarized) values contained within data sets. For instance, using our framework, it is possible to express queries that range from simple "schema-level" filters such as "*Find all data sets that contain height measurements of trees within experimental locations*" to queries that access, summarize, and select results based on the values within data sets such as "*Find all data sets that have trees with a maximum height measurement larger than 20* m *within experimental locations having an area smaller than 10* m^2".

Finally, we describe different storage and query evaluation approaches that have been implemented to support the framework. We consider both a "data warehouse" approach that uses a single "materialized" database to store underlying observational data sets

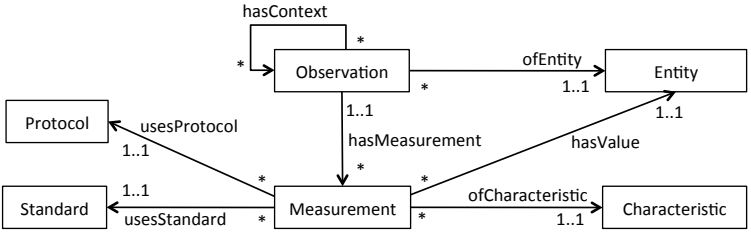


Fig. 2. Main observational modeling constructs used in semantic annotation and data discovery

(where query evaluation involves rewriting a discovery query into a query over the warehouse) and an approach that treats semantic annotations as logical views over the underlying data set schemas (where query evaluation involves rewriting the original query using the annotation into corresponding queries over the underlying data sets). Based on our initial experimental results, we demonstrate the feasibility of querying a large corpus using these approaches, and that querying data in place can lead to better performance compared with more traditional warehousing approaches.

The rest of this paper is organized as follows. In Sec. 2 we present the observational model, semantic annotation language, and data-discovery language used within our framework. In Sec. 3 we describe two implementation approaches. In Sec. 4 we present an initial experimental evaluation. In Sec. 5 we discuss related work, and in Sec. 6 we summarize our contributions.

2 Semantic Annotation and Discovery Framework

Fig. 2 shows the modeling constructs we use to describe and (depending on the implementation) store observational data. An *observation* is made of an *entity* (e.g., biological organisms, geographic locations, or environmental features, among others) and primarily serves to group a set of measurements together to form a single "*observation event*". A *measurement* assigns a value to a *characteristic* of the observed entity (e.g., the height of a tree), where a value is denoted through another entity (which includes primitive values such as integers and strings, similar to pure object-oriented models). Measurements also include *standards* (e.g., units) for relating values across measurements, and can also specify additional information including collection protocols, methods, precision, and accuracy (not all of which are shown in Fig. 2 due to space limitation). An observation (event) can occur within the *context* of zero or more other observations. Context can be viewed as a form of dependency, e.g., an observation of a tree specimen may have been made within a specific geographic location, and the geographic location provides important information for interpreting and comparing tree measurements. In this case, by establishing a context relationship between the tree and location observations, the measured values of the location are assumed to be constant with respect to the measurements of the tree (i.e., the tree measurements are dependent on the location measurements). Context forms a *transitive* relationship among observations. Although not

considered here, we also employ a number of additional structures in the model for representing complex units, characteristics, and named relationships between observations and entities [9]. When describing data sets using the model of Fig. 2, domain-specific entity, characteristic, and standard classes are typically used. That is, our framework allows subclasses of the classes in Fig. 2 to be defined and related, and these terms can then be used when defining semantic annotations.

A key feature of the model is its ability for users to assert properties of entities (as measurement characteristics or contextual relationships) without requiring these properties to be interpreted as *inherently* (i.e., *always*) true of the entity. Depending on the context an entity was observed (or how measurements were performed), its properties may take on different values. For instance, the diameter of a tree changes over time, and the diameter value often depends on the protocol used to obtain the measurement. The observation and measurement structure of Fig. 2 allows RDF-style assertions about entities while allowing for properties to be contextualized (i.e., the same entity can have different values for a characteristic under different contexts), which is a crucial feature for modeling scientific data [9]. Although shown using UML in Fig. 2, the model has been implemented (together with a number of domain extensions) using OWL-DL.[1]

2.1 Semantic Annotation

Semantic annotations are represented using a high-level annotation language (e.g., see Fig. 3), and each annotation consists of two separate parts: (1) a *semantic template* that defines specific observation and measurement types (and their various relationships) for the data set; and (2) a *mapping* from individual attributes of the data set to measurement types defined within the semantic template. The left side of Fig. 3 gives an example annotation for the first table of Fig. 1. Here we define four observation types denoting measurements of sites, plots, soils, and trees, respectively. A site observation contains a simple (so-called "nominal") measurement that gives the name of the site. Similarly, a plot observation records the name of the plot (where a plot is used as an experimental replicate) as well as the plot area. Here, plots are observed within the context of a corresponding site. A soil observation consists of an acidity measurement and is made within the context of a plot observation (although not shown, we would typically label the context relation in this case to denote that the soil is part of the plot). Finally, a tree observation consists of the taxonomic name of the tree along with height and diameter measurements in meters and centimeters, respectively.

The right side of Fig. 3 shows the relationship between (a portion of) the semantic template (top) and an attribute mapping (dashed-lines, middle) from the underlying data set schema (bottom) to the template. As shown, each attribute is assigned to a single measurement type in the template. This approach follows the typical view of attributes in data sets as specifying measurements, where the corresponding entities, observation events, and context relationships are implied by the template. We note that users will not typically specify annotations directly using the syntax shown in Fig. 3. Instead, we have developed a graphical user-interface within [2] that allows users to specify attribute-level mappings to measurement types and the corresponding measurement and

[1] e.g., see http://ecoinformatics.org/oboe/oboe.1.0/oboe-core.owl

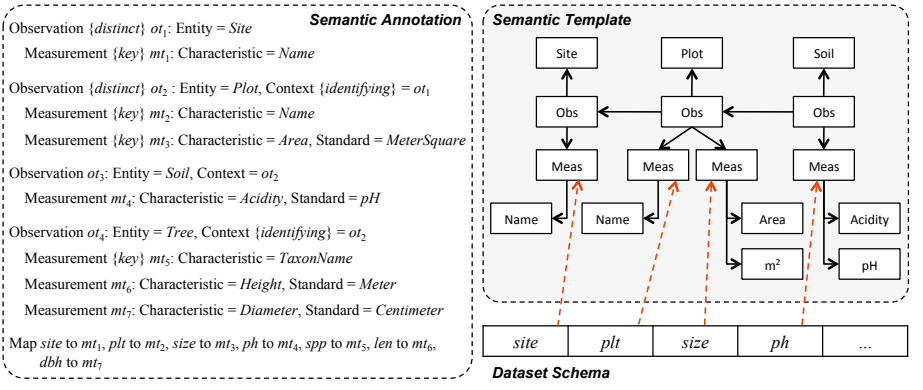


Fig. 3. Semantic annotation of the first data set of Fig. 1 showing the high-level annotation syntax (left) and a portion of the corresponding "semantic template" and schema mapping (right)

observation types of the data set. The annotation language shown here is used to store (via an XML serialization) the mappings and semantic templates generated by the tool.

The meaning of a semantic annotation can be viewed as the result of processing a data set row-by-row such that each row creates a valid instance of the semantic template. For example, in the first row of the data set, the site value "GCE6" implies: (1) an instance m_1 of the measurement type mt_1 whose value is "GCE6"; (2) an instance c_1 of the Name characteristic for m_1; (3) an instance o_1 of an observation (corresponding to type ot_1) having measurement m_1; and (4) an instance e_1 of the Site entity such that e_1 is the entity of o_1. Similarly, assuming the plot attribute value "A" of the first row corresponds to an observation instance o_2 (of observation type ot_2), the context definition for ot_2 results in o_2 having o_1 as context.

The "key", "identifying", and "distinct" constraints are used to further specify the structure of semantic-template instances. These constraints are similar to key and weak-entity constraints used within ER models. If a measurement type is defined as a *key* (e.g., mt_1 in Fig. 3), then the values of instances for these measurement types identify the corresponding observation entity. For example, both the first and second row of the first table in Fig. 1 have a site value of "GCE6". Thus, if the observation instance for the site attribute in the second row of the table is o_3, then the key constraint of mt_1 requires that e_1 be an entity of o_3 where e_1 is the entity instance of the first-row's corresponding observation. An *identifying* constraint requires the identity of one observation's entity to depend (through context) on the identity of another observation's entity. In our example, plot names are unique only within a corresponding site. Thus, a plot with the name "A" in one site is not the same plot as a plot with the name "A" in a different site. Identifying constraints define that the identity of an observation's entity is determined by both its own key measurements and its identifying observations' key measurements. Thus, each site name determines the entity observed through the key constraint, whereas, each plot name determines the plot entity with respect to both its name and its corresponding site (as given by the identifying constraint on the context relationship). The "distinct" constraint on observations is similar to the "key" constraint on measurements, except that it is used to uniquely identify observations (as opposed to observation entities). In

Fig. 3, each row with the same value for the site attribute maps not only to the same observed entity (via the key constraint) but also to the same observation instance (via the distinct constraint). A distinct constraint can only be used if each measurement of the observation is constrained to be a key.

More formally, we can *materialize* semantic annotations as follows. First, we represent sets of annotations using the following relations.

- *Annot*(a, d) states that a is an annotation of data set with id d.
- *ObsType*(a, ot, et, *isDistinct*) states that ot is an observation type in annotation a, has entity type et, and whether it distinct.
- *MeasType*(a, mt, ot, ct, st, ..., *isKey*) states that mt is a measurement type in a, is for observation type ot, and has characteristic type ct, standard type st, etc., and whether mt is defined as a key.
- *ContextType*(a, ot, ot', *isId*) states that observation type ot' is a context type of observation type ot in a, and whether the context relationship is identifying.
- *Map*(a, $attr$, mt, ϕ, v) states that data set attribute $attr$ is mapped to measurement type mt in a, where ϕ is an optional condition specifying whether the mapping applies (based on the values of attributes within the data set) and v is an optional value to use for the measurement (instead of the data set value).

We use the following relations to represent instances of semantic templates.

- *Entity*(d, e, et) states that entity e in data set d is an instance of entity type et.
- *Obs*(d, o, ot, e) states that observation o in data set d is of type ot and is an observation of entity e.
- *Meas*(d, m, mt, v, o) states that measurement m in d is of measurement type mt, has the value v, and is a measurement for observation o.
- *Context*(d, o, o') states that observation o is within the context of o' in d.

We can then evaluate the mapping defined by a semantic annotation a over a data set d using the following algorithm, which results in populating the above relations for template instances.

Algorithm MaterializeDB(a, d)

1). $EntityIndex = \emptyset$; // an index of the form $\{\langle ot, keyvals\rangle \rightarrow e\}$
2). **for each** $row = \langle attr_1, attr_2, \cdots, attr_n\rangle \in d$
3). $MeasSet =$ CreateMeasurements(a, row);
4). // partition measurements based on observation types
5). $MeasIndex =$ PartitionMeasurements(a, $MeasSet$); // returns index $\{ot \rightarrow \{m\}\}$
6). $ObsIndex = \emptyset$; // an index of the form $\{ot \rightarrow o\}$
7). **for each** $ot \rightarrow \{m\} \in MeasIndex$
8). $e =$ CreateEntity(a, ot, $\{m\}$, $EntityIndex$); // updates $EntityIndex$
9). CreateObservation(a, ot, e, $ObsIndex$); // updates $ObsIndex$
10). ConnectContext(a, $ObsIndex$);

As shown, while processing each row we create measurement instances for each mapped attribute (cell) in the row (Line 3), link them to their related observation instances (Line 5–9), and then create proper context links between observation instances (Line 10). The *EntityIndex* is used to ensure only unique entities are created within the data set (based on the values for measurements having a key constraint). Thus, before an entity instance

is created (Line 8), the CreateEntity function first checks using the index if the entity has already been created from a previous row. The CreateMeasurements, CreateObservation, and ConnectContext functions are straightforward and each use the annotation's semantic template to create and connect the corresponding instances. This algorithm runs in $O(n \log m)$ time where n is the number of rows in a data set and m ($\ll n$) is the number of distinct keys within the data set. The algorithm uses $O(nc)$ space where c is the number of columns in the data set (thus, nc is the total number of cells).

The semantic annotation language can easily be expressed using standard schema mapping approaches [16], i.e., annotations have a straightforward reduction to source-to-target tuple-generating dependencies and target equality-generating dependencies. A source-to-target tuple-generating dependency (*st-tdg*) is a first-order formula of the form $\forall \bar{x}(\varphi(\bar{x}) \rightarrow \exists \bar{y}\, \psi(\bar{x}, \bar{y}))$ where $\varphi(\bar{x})$ and $\psi(\bar{x}, \bar{y})$ are conjunctions of relational atoms over source and target schemas, respectively, and $\bar{x}$ and $\bar{y}$ are tuples of variables. We can use st-tdgs to define instances of semantic templates, e.g., the following rule maps the first attribute in the data set of Fig. 3 to measurement type mt_1, where R is used as the name of the data set relation:

$$\forall \bar{x}(R(\bar{x}) \rightarrow \exists \bar{y}\, \mathrm{Meas}(\mathtt{d}, y_1, \mathtt{mt}_1, x_1, y_2) \wedge \mathrm{Obs}(\mathtt{d}, y_2, \mathtt{ot}_1, y_3) \wedge \mathrm{Entity}(\mathtt{d}, y_3, \mathtt{Site}))$$

Here we assume x_1 is the first variable in $\bar{x}$, each y_i in the rule is a variable of $\bar{y}$, and $\mathtt{d}$, $\mathtt{mt}_1$, $\mathtt{ot}_1$, and $\mathtt{Site}$ are constants. A target equality-generating dependency (*t-egd*) takes the form $\forall \bar{y}(\phi(\bar{y}) \rightarrow u = v)$ where $\phi(\bar{y})$ is a conjunction of relational atoms over the target schema and u, v are variables in $\bar{y}$. We can use t-egds to represent key, identifying, and distinct constraints, e.g., the following rule can be used to express the key constraint on measurement type mt_1:

$$\begin{aligned}\forall \bar{y}(&\mathrm{Meas}(\mathtt{d}, m_1, \mathtt{mt}_1, v, o_1) \wedge \mathrm{Obs}(\mathtt{d}, o_1, \mathtt{ot}_1, e_1) \wedge \mathrm{Meas}(\mathtt{d}, m_2, \mathtt{mt}_1, v, o_2) \\ &\wedge \mathrm{Obs}(\mathtt{d}, o_2, \mathtt{ot}_1, e_2) \rightarrow e_1 = e_2)\end{aligned}$$

The annotation language we employ is also similar to a number of other high-level mapping languages used for data exchange (e.g., [10,6]), but supports simple type associations to attributes (e.g., as shown by the red arrows on the right of Fig. 3) while providing well-defined and unambiguous mappings from data sets to the observation and measurement schema.

2.2 Data Discovery Queries

Data discovery queries can be used to select relevant data sets based on their observation and measurement types and values. A *basic discovery query* Q takes the form

$$Q ::= \mathtt{EntityType}(\mathtt{Condition})$$

where `EntityType` is a specific observation entity class and `Condition` is a conjunction or disjunction of zero or more conditions of the form

$$\begin{aligned}\mathtt{Condition} \;::=\;& \mathtt{CharType}\;[\;\mathtt{op}\;\mathtt{value}\;[\;\mathtt{StandardType}\;]\;] \\ \mid\;& f(\mathtt{CharType})\;[\;\mathtt{op}\;\mathtt{value}\;[\;\mathtt{StandardType}\;]\;] \\ \mid\;& \mathtt{count}(\;[\;\mathtt{distinct}\;]\;*)\;\mathtt{op}\;\mathtt{value}\end{aligned}$$

Square brackets above denote optional components and f denotes a standard aggregation function (i.e., sum, avg, min, or max). A data set is returned by a basic discovery query if it contains observations of the given entity type that satisfy the corresponding conditions. We consider three basic types of conditions (for the three syntax rules above): (1) the observation must contain at least one measurement of a given characteristic type (`CharType`) with a measured value satisfying a relational (i.e., $=$, $\neq$, $>$, $<$, $\geq$, $\leq$) or string comparison (e.g., `contains`); (2) the aggregate function applied to measurements of the characteristic type (`CharType`) for all observations of the entity type must satisfy the relational comparison; and (3) the number (`count`) of all observations of the entity type must satisfy the relational comparison (where `distinct` restricts the set of observations to those within unique entities). For instance, in the following basic discovery queries

Tree(TaxonName = 'piru')
Tree(TaxonName = 'piru' $\wedge$ count(distinct $*$) ≥ 5)

the first query select data sets with at least one Tree observation labeled as having the (abbreviated) taxon name "piru", and the second query restricts the returned data sets of the first query to contain at least five such observations.

A *contextualized discovery query* generalizes basic discovery queries to allow selections on context. A contextualized query Q_C for $n \geq 1$ has the form

$$Q_C \ ::= \ Q_1 \rightarrow Q_2 \ \cdots \rightarrow Q_n$$

where each Q_i is a basic discovery query and $\rightarrow$ denotes a context relationship. In particular, a data set satisfies a contextualized query if it satisfies each basic query Q_i and each matching Q_i is related by the given context constraint. To illustrate, the following examples can be used to express the two queries of Sec. 1:

Tree(Height) $\rightarrow$ Plot()
Tree(max(height) $\geq$ 20 Meter) $\rightarrow$ Plot(area $<$ 10 MeterSquared).

That is, the first query returns data sets that contain height measurements of trees within plots (experimental locations), and the second returns data sets that have trees of a maximum height larger than 20 m within plots having an area smaller than 10 m^2. For a collection of data sets D and a contextualized discovery query Q, in the normal way, we write $Q(D)$ to denote the subset of data sets in D that satisfy Q. Note that $Q(D)$ can be computed on a per data set basis, i.e., by checking each data set in D individually to see whether it satisfies Q.

3 Implementation Strategies

In this section we describe two different strategies for evaluating data discovery queries over annotated observational data sets, as shown in Fig. 4. Both strategies utilize semantic annotations (and corresponding ontologies) to answer discovery queries. We assume annotations are stored using the relations described in Sec. 2.1, namely the *Annot*, *ObsType*, *MeasType*, *ContextType*, and *Map* tables. The two approaches differ in how they utilize different representations of the underlying data sets. The first query strategy (*rdb*)

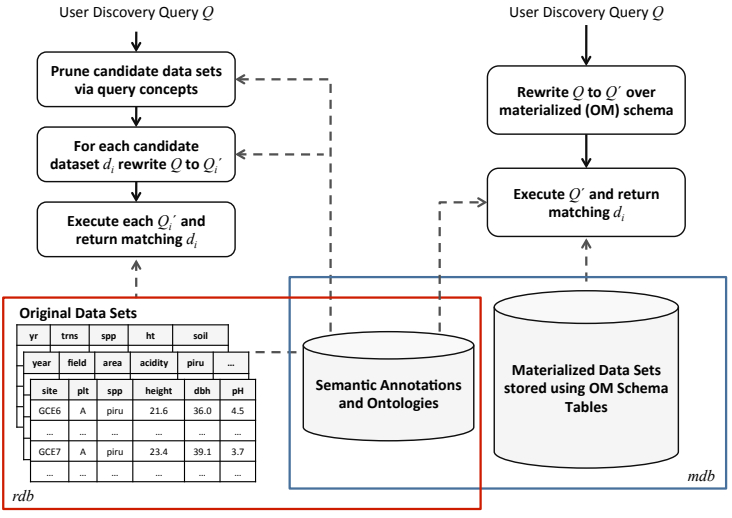


Fig. 4. Different strategies for answering a data discovery query Q: the first strategy stores each data set in its "raw" form (left), and the second strategy materializes each data set into the a common schema (right).

stores each data set in its "raw" form (i.e., "in place") according to its defined schema. Thus, each data set is stored as a distinct relation in the database. For instance, both data sets of Fig. 1 would be stored as tables without any changes to their schemas. The second query strategy (*mdb*) materializes each data set using the *MaterializeDB* algorithm. In this approach, the observations and measurements stored in each data set are represented using the *Entity*, *Obs*, *Meas*, and *Context* tables described in Sec. 2.1.

3.1 Query Evaluation over In-Place Database

Evaluating a discovery query Q in the *rdb* approach consists of three steps. Here we assume that each data set (denoted by id d) is stored in a relation R_d. The first step prunes the search space of candidate data sets to select only those data sets with the required entity, characteristic, and standard types specified within Q. The candidate data sets (i.e., those that potentially match Q) are selected by accessing the semantic-annotation relations $ObsType$, $MeasType$, and $Annot$. This pruning step can help decrease the cost of evaluating discovery queries by reducing the number of data sets whose values must be accessed (for those queries that select data sets based on data values). The second step translates Q into an SQL query Q' over each candidate data set relation R_d. After the first step, we obtain the measurement types (*mt*) related to the discovery query Q (via the *MeasType* relation). We then find the attributes $attr_q$ in each candidate relation R_d via the Map relation (which contains correspondences between attributes $attr$ and measurement types mt). If a basic query does not have any aggregations, these attributes are enough to form the resulting SQL over R_d. Otherwise, we must obtain the key measurement types $attr_{key}$ for the observation types of Q. The key measurement types of an observation type consist of its own key measurement types

and those of its identifying observation types, which must be retrieved by traversing the identifying context chain (i.e., by searching the $ObsType$ and $ContextType$ relations). Once the needed attributes for each candidate data set d are obtained, Q is translated into the following SQL query (where square brackets denote optional clauses).

```
SELECT DISTINCT d[, attr_key] FROM R_d
WHERE non-aggregation conditions
[GROUP BY attr_key]
[HAVING aggregation condition];
```

We illustrate the rewriting process with the following example. Consider the semantic annotations in Fig. 3 and the basic discovery query

$$\text{Tree(TaxonName = 'piru'} \wedge \text{count(distinct} *) \geq 5)$$

from Sec. 2.2. The first step is to find the attributes involved in the condition (i.e., TaxonName = 'piru'). From the entity type "Tree" and the characteristic "TaxonName", we can find the corresponding observation type "ot_4", measurement type "mt_5", and the attribute "spp". We then find the key attributes to perform the aggregation. The key measurement types for entity type "Tree", whose observation type is ot_4, come from ot_4 and ot_4's identifying observation types ot_2 and ot_1. Since these observation types' key measurement types are mt_5, mt_3, mt_2 and mt_1, the key attributes $attr_{key}$ are spp, $size$, plt, and $site$ (from the Map relation). The resulting SQL query is expressed as follows where the data set id is denoted d_1 and corresponding relation is denoted R_{d1}.

```
SELECT DISTINCT d1 FROM R_d1
WHERE spp = 'piru'
GROUP BY d1, spp, size, plt, site
HAVING count(*) ≥ 5
```

Finally, in the third step of the *rdb* approach, we execute the SQL query for each of the candidate data sets such that the answer for Q is the union of these results.

Cost analysis. The major computation cost using the *rdb* approach is to send multiple SQL queries to the database server to search the needed information for all the different candidate data tables. Thus, the cost increases with larger numbers of candidate data sets.

3.2 Query Evaluation over Materialized Database

The second query strategy (*mdb*) evaluates a given discovery query Q over the materialized database by directly rewriting Q into an SQL query expressed over the annotation and instance relations of Sec. 2.1. This approach differs from *rdb*, which requires obtaining the underlying attributes for *each* individual candidate table and where one SQL query is constructed per candidate table. Instead, using *mdb*, a single SQL query is created to answer the entire basic data discovery query.

The way in which a basic query is rewritten depends on whether it has an aggregation condition. For a basic query without an aggregation condition, we access the $Entity$ and Obs relations to check the entity conditions, and search the *MeasType* and *Meas* tables for characteristic and standard conditions. For a basic discovery query with an aggregation condition, we perform the aggregation by grouping $Entity.e$ or $Obs.o$

depending on whether the aggregation is on a distinct entity instance or observation instance. Thus, a discovery query Q can be re-written to an SQL query Q' using the *mdb* approach as follows.

```
SELECT DISTINCT Annot.d [, Entity.e][, Obs.o]
FROM Annot, Entity, Obs, MeasType, Meas
[WHERE table join condition [AND selection condition]]
[GROUP BY Annot.d[, Entity.e][, Obs.o] ]
[HAVING aggregation condition ];
```

where:

```
table join condition= (Annot.a = MeasType.a) AND (Annot.d = Meas.d)
  AND (MeasType.mt=Meas.mt) AND(Meas.o=Obs.o) AND (Obs.e=Entity.e)
```

If a basic query contains multiple measurement conditions (of the same entity type), the corresponding SQL query must be combined using "INTERSECTION" or "UNION" operations to answer the basic query of one entity type. For example, the query

Tree(TaxonName = 'piru' $\wedge$ count(distinct $*$) ≥ 5)

is rewritten using the *mdb* approach as:

```
SELECT DISTINCT Annot.d
FROM Annot, Entity, Obs, MeasType, Meas
WHERE table join condition
    AND MeasType.ct='TaxonName' AND Meas.v = 'piru'
GROUP BY Annot.d, Obs.o
HAVING COUNT(*)>5;
```

Cost analysis: The major computation cost in *mdb* involves the cost of joining over the type and instance relations, and the selection cost over the measurement values.

De-normalized materialized database. The join condition shows that a large portion of the cost comes from the join operation over the measurement instance, observation instance, and entity instance relations. To reduce the join cost, we can de-normalize the instance relations $Entity$, Obs, and $Meas$ into a single relation. This strategy will use slightly more space, but can improve the performance of query evaluation for *mdb*.

Horizontally partitioned database. When all the data values are placed into a single measurement table, their data types must be of the same type. This requires using type casting functions provided by the database system to perform type conversion for evaluating queries with algebraic or aggregation operators. This incurs a full scan of the $Meas$ table regardless of whether there is an index on the value columns or not. In our current implementation, we address this issue by partitioning the measurement instance table according to the different data types (e.g., numeric, char, etc.). This partitioning does not incur additional space overhead.

3.3 Executing Complex Discovery Queries

To evaluate more complex data-discovery queries that consist of context relationships, or conjunctions and disjunctions of basic queries (with different entity types), we decompose the query into query blocks and then combine the results of each decomposed

query block. In the first step of query decomposition, a complex query is reformulated into disjunctive normal form (DNF) such that each DNF component is either (1) a basic data discovery query, (2) a conjunction of basic discovery queries with different entity types, or (3) a contextualized discovery query. In the second step of implementing and integrating the decomposed query blocks, we propose two approaches, ExeD and ExeH.

ExeD: Executing a query block based on Decomposed query units. In ExeD, each DNF is further *decomposed* into basic discovery query components. Each of these most decomposed query units is rewritten and executed using one of the strategies discussed in the above two sections. Then, the result of each such most decomposed component is combined (outside of the DBMS) to obtain the result of every DNF. In particular, for the first case where the DNF clause is a basic discovery query, the algorithm simply executes this basic query. For the second case where the DNF component is a conjunction of basic discovery queries, the algorithm intersects the results of the further decomposed components in this DNF. For the third case of contextualized discovery queries, the algorithm runs the basic queries and their context queries and intersects the results. Finally, the results of different DNF components are unioned as the final result.

ExeH: Executing a query block based on Holistic sub-queries. The ExeD approach may incur unnecessarily repeated scans of the database since it evaluates each decomposed basic unit using the DBMS and combines the results externally, outside of the system. For instance, consider the query

$$\text{Tree(max(height)} \geq 20 \text{ Meter)} \rightarrow \text{Plot(area} < 10 \text{ MeterSquared)}.$$

Using the ExeD approach, we need to send two basic queries, Tree(max(height) $\geq$ 20 Meter) and Plot(area $<$ 10 MeterSquared), to the same table. Instead, we form a "holistic" SQL query for each possible basic query block. This holistic SQL is then executed by taking advantage of the optimization capabilities of the DBMS.

Note that not every complex query can be rewritten to one holistic SQL query. Specifically, queries with aggregations must be performed by grouping key measurements. When we have discovery queries with multiple aggregation operations, the *group by* attributes for each aggregation may not be the same. In ExeH, we categorize query blocks into those with and without aggregations. All the query blocks without aggregation conditions are combined and rewritten into one holistic SQL query, while the query blocks with aggregations are processed individually.

4 Experimental Evaluation

In this section we describe the results of our experimental evaluation of the framework and algorithms discussed above. Our implementation was written in Java, and all experiments were run using an iMac with a 2.66G Intel processor and 4G virtual memory. We used PostgreSQL 8.4 as the back-end database system. To report stable results, all numbers in our figures represent the average result of 10 different runs (materialization tasks or queries) with the same settings for each case.

Data. We generated synthetic data to simulate a number of real data sets used within the Santa Barbara Coastal (SBC) Long Term Ecological Research (LTER) project [4].

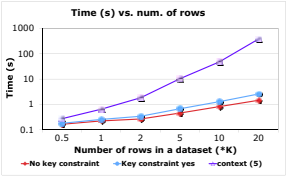


Fig. 5. Data materialization

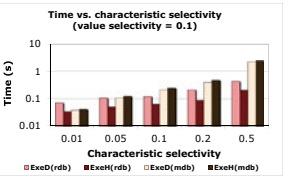


Fig. 6. Time vs. cha. selectivity

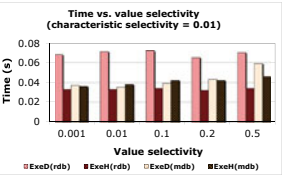
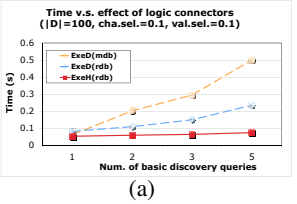
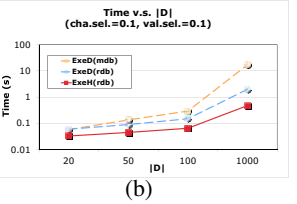


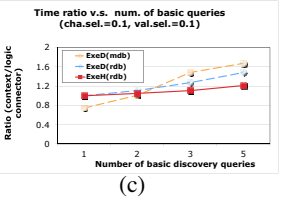
Fig. 7. Time vs. val. selectivity



(a)



(b)



(c)

Fig. 8. Scalability test on (a) the number of logic connectors (e.g., AND and OR), (b) data set size (with three basic queries connected by the logic connector)), and (c) length of context chains (compared with logic connectors). (ExeH(mdb) is not included because it shows similar results to ExeD(mdb).)

This repository contains ∼130 data sets where each one has 1K to 10K rows and on average 15 to 20 columns. To simulate this repository (to test scalability, etc.), our data generator used the following parameters. The average number of attributes and records in a data set is 20 and 5K respectively. The average number of characteristics for an entity is two. The distinctive factor $f \in (0, 1]$, which represents the ratio of distinct entity/observation instances in a data set, is set to 0.5. Our synthetic data generator also controls the *attribute selectivity* to facilitate the test of query selectivity. In particular, given a selectivity $s \in (0, 1]$ of an attribute $attr$, a repository with $|D|$ data sets will be generated to have $|D| \cdot s$ data sets with attribute $attr$.

Queries. Test queries were generated with controlled characteristic selectivity and value selectivity, where the characteristic selectivity determines the percentage of data sets that satisfy a given query and the value selectivity determines the percentage of data rows in a data set satisfying the query. We omit the details of query generation due to space limitations.

Test 1: Materializing data. We first tested the efficiency of the materialization method using data sets generated with distinctive factor $f = 0.5$, number of columns 20, and various number of rows. The annotations over these data sets are the same on observation and measurement types. They differ in the key constraints. "No key constraint" and "Key constraint yes" refer to cases where either no key or key constraints exist in the semantic annotations of data sets; both of these two cases do not include any context constraints. The "context (5)" represents data sets that are semantically annotated with a context chain of 5 observation types (with implicit key constraints).

Fig. 5 shows that the materialization time for each case (every line) is linear to the number of rows in these data sets. This is consistent with our analysis in Sec. 2.1. The

"no key constraint" uses the least amount of time because it does not need to do any additional computation to enforce the uniqueness of the entity and observation instances. The "context (5)" uses the most amount of time because of the context materialization.

Test 2: Querying databases. We also tested the effectiveness and efficiency of our two query strategies: query evaluation using rdb and mdb by utilizing *ExeD* and *ExeH*. Test 2.1 examines how the different query strategies are affected by *value selectivity and characteristic selectivity*. For this test, we used a data repository with 100 data sets and evaluated queries consisting of two basic discovery units connected with the "AND" logic connector. Fig. 6 shows that the execution time of different query strategies increases with the increase of the characteristic selectivity. For *rdb*, higher characteristic selectivity means that more candidate tables are involved in answering a query, thus more SQL queries (against these candidates) are executed. For *mdb*, higher selectivity involves more materialized instances in the join condition, thus more time is used. When we fix the characteristic selectivity and vary the value selectivity, we can see that the execution time is almost constant (Fig. 7) for *rdb* because the number of candidate data tables is the same. For *mdb*, the execution time grows slightly with the increase of the value selectivity also due to the increase in the number of materialized instances. These two figures show that, with smaller characteristic selectivity (e.g., 0.01), the query strategy over *mdb* performs better than for *rdb*. But with larger characteristic selectivities (e.g., 0.5), query strategies over *rdb* perform better. This is because queries over larger amounts of materialized instances, which is the case for larger characteristic selectivities, take more time to perform joins, compared with executing SQL queries over individual candidate data tables in *rdb*.

Test 2.2 focuses on the scalability of the different query strategies. Fig. 8(a) illustrates that the three methods grow linearly to the number of logic connectors. When the number of logic connectors grow, *ExeD(mdb)* grows much faster than *Exe(rdb)*. This is because every basic query in the complex query needs to access the large number of instance tables once for *mdb*. While using *rdb*, when the number of logic connectors is large, *ExeH(rdb)* is still almost constant because the times required to scan the database are almost the same. Fig. 8(b) shows that these different approaches grow linearly to the size of data sets $|D|$ when the value selectivity and characteristic selectivity are fixed. Fig. 8(c) plots the ratio of the execution time of performing a complex query with logic connectors and that of performing a contextualized query with the same number of basic query units. All three methods grow linearly to the number of basic queries. However, the query over *rdb* grows slower than the *mdb* since the latter needs to access the context instance table. As these results show, rewriting queries to the underlying data set schemas outperforms the materialization approach (i.e., the standard warehousing approach) with respect to both the cost of storage (especially due to de-normalization and executing the materialization algorithm) and overall query execution time.

5 Related Work

Data management systems are increasingly employing annotations to help improve search (e.g., [12, 18, 8]). For example, MONDRIAN [12] employs an annotation model and query operators to manipulate both data and annotations. However, users must be

familiar with the underlying data structures (schemas) to take advantage of these operators, which is generally not feasible for observational data in which data sets exhibit a high degree of structural and semantic heterogeneity. Efforts have also been carried out for leveraging annotations, e.g., for the discovery of domain-specific data [13,20]. These approaches are largely based on keyword queries, and do not consider structured searches. Our work differs from these approaches in that we consider a highly structured and generic model for annotations with the aim of providing a uniform approach for issuing structured data-discovery searches. Our work is closely aligned to traditional data integration approaches (e.g., [14,16]), where a global mediated schema is used to (physically or logically) merge the structures of heterogeneous data sources using mapping constraints among the source and target schemas. As such, the observational model we employ in our framework can be viewed as a (general-purpose) mediation schema for observational data sets. This schema can be augmented with logic rules (as target constraints) and uses the semantic annotations as mapping constraints. However, instead of users specifying logic constraints directly, we provide a high-level annotation language that simplifies the specification of mappings and more naturally aligns with the observation model. In addition, our work focuses on implementing practical approaches for rewriting and optimizing queries (that can include aggregation and summarization operators) over our annotation approach. In particular, our goal is to create a feasible, scalable, and deployable system for applying these approaches for data discovery and exploratory data analysis within existing scientific data repositories.

6 Conclusion

We have presented a novel framework for querying observational data based on formal semantic annotations and a data discovery language that allows structural queries over both schema and data. We also have considered different strategies for efficiently implementing the framework. Our approach involves different forms of query rewriting over annotations and data set schemas. We also have examined the effect of different storage schemas on the query strategies. Our experiments show that in most cases answering queries "in place" outperforms more traditional warehouse-based approaches. As future work we intend to continue to investigate approaches for optimization including the use of indexing schemes and their use in query rewriting approaches.

References

1. Knowledge network for biocomplexity (KNB), http://knb.ecoinformatics.org
2. Morpho metadata editor, http://knb.ecoinformatics.org
3. OpenGIS: Observations and measurements encoding standard (O&M), http://www.opengeospatial.org/standards/om
4. Santa Barbara Coastal LTER repository, http://sbc.lternet.edu/data
5. The Digital Archaeological Record (tDAR), http://www.tdar.org
6. An, Y., Mylopoulos, J., Borgida, A.: Building semantic mappings from databases to ontologies. In: AAAI (2006)
7. Berkley, C., et al.: Improving data discovery for metadata repositories through semantic search. In: CISIS, pp. 1152–1159 (2009)

8. Bhagwat, D., Chiticariu, L., Tan, W.C., Vijayvargiya, G.: An annotation management system for relational databases. In: VLDB (2004)
9. Bowers, S., Madin, J.S., Schildhauer, M.P.: A conceptual modeling framework for expressing observational data semantics. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 41–54. Springer, Heidelberg (2008)
10. Fagin, R., Haas, L.M., Hernández, M., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema mapping creation and data exchange. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 198–236. Springer, Heidelberg (2009)
11. Fox, P., et al.: Ontology-supported scientific data frameworks: The virtual solar-terrestrial observatory experience. Computers & Geosciences 35(4), 724–738 (2009)
12. Geerts, F., Kementsietsidis, A., Milano, D.: Mondrian: Annotating and querying databases through colors and blocks. In: ICDE, p. 82 (2006)
13. Güntsc, A., et al.: Effectively searching specimen and observation data with TOQE, the thesaurus optimized query expander. Biodiversity Informatics 6, 53–58 (2009)
14. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: the teenage years. In: VLDB 2006 (2006)
15. Balhoff, J., et al.: Phenex: Ontological annotation of phenotypic diversity. PLoS ONE 5 (2010)
16. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: PODS 2005 (2005)
17. Pennings, S., et al.: Do individual plant species show predictable responses to nitrogen addition across multiple experiments? Oikos 110(3), 547–555 (2005)
18. Reeve, L., Han, H.: Survey of semantic annotation platforms. In: SAC 2005 (2005)
19. Sorokina, D., et al.: Detecting and interpreting variable interactions in observational ornithology data. In: ICDM Workshops, pp. 64–69 (2009)
20. Stoyanovich, J., Mee, W., Ross, K.A.: Semantic ranking and result visualization for life sciences publications. In: ICDE, pp. 860–871 (2010)

A Scalable Tag-Based Recommender System for New Users of the Social Web

Valentina Zanardi and Licia Capra

Dept. of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK
{V.Zanardi,L.Capra}@cs.ucl.ac.uk

Abstract. Folksonomies have become a powerful tool to describe, discover, search, and navigate online resources (e.g., pictures, videos, blogs) on the Social Web. Unlike taxonomies and ontologies, which overimpose a hierarchical categorisation of content, folksonomies empower end users, by enabling them to freely create and choose the categories (in this case, tags) that best describe a piece of information. However, the freedom afforded to users comes at a cost: as tags are informally defined and ungoverned, the retrieval of information becomes more challenging. In this paper, we propose *Clustered Social Ranking* (CSR), a novel search and recommendation technique specifically developed to support new users of Web 2.0 websites finding content of interest. The observation underpinning CSR is that the vast majority of content on Web 2.0 websites is created by a small proportion of users (*leaders*), while the others (*followers*) mainly browse such content. CSR first identifies who the leaders are; it then clusters them into communities with shared interests, based on their tagging activity. Users' queries (be them searches or recommendations) are then directed to the community of leaders who can best answer them. Our evaluation, conducted on the CiteULike dataset, demonstrates that CSR achieves an accuracy that is comparable to the best state-of-the-art techniques, but at a much smaller computational cost, thus affording it better scalability in these fast growing settings.

Keywords: cold-start problem, scalability, recommender systems, social tagging, clustering.

1 Introduction

The rise of Web 2.0 has transformed users from passive consumers to active producers of content. This has exponentially increased the amount of information that is available to users, from videos on sites like YouTube and MySpace, to pictures on Flickr, music on Last.fm, blogs on Blogger, and so on. This content is no longer categorised according to pre-defined taxonomies (or ontologies). Rather, a new trend called social (or folksonomic) tagging has emerged, and quickly become the most popular way to describe content within Web 2.0

A. Hameurlain et al. (Eds.): DEXA 2011, Part I, LNCS 6860, pp. 542–557, 2011.

websites. Unlike taxonomies, which overimpose a hierarchical categorisation of content, folksonomies empower end users by enabling them to freely create and choose the tags that best describe a piece of information (a picture, a blog entry, a video clip, etc.). However, this freedom comes at a cost: since tags are informally defined, continually changing, and ungoverned, *finding* content of interest has become a main challenge, because of the number of synonyms, homonyms, polysemy, as well as the inevitable heterogeneity of users and the noise they introduce.

In order to assist users finding content of their own interest within this information abundance, new techniques, inspired by traditional recommender systems, have been developed: users' profiles are built, collecting information about their tastes/interests; these profiles are then processed to predict what resources they will like. While high accuracy can be afforded for users whose preferences are well known in the system (e.g., users who have rated/tagged a lot of content) [9], very little has been done so far for new users. However, the so-called *cold start* problem is dominant in Web 2.0 websites, where a large number of new users joins the system daily; furthermore, their tastes are more difficult to learn than in traditional recommender systems, as they are not simply expressed as numerical ratings over consumed content, but as freely chosen sets of tags associated to it. In order to retain these users, a recommender system must be capable of recommending content, even when very little information is available about a user's interests.

In this paper, we propose *Clustered Social Ranking* (CSR), a novel search and recommendation technique specifically developed to support new users of Web 2.0 websites finding content of interest. The observation underpinning CSR is that the vast majority of content on Web 2.0 websites is created by a small proportion of users, while the others mainly browse such content. We call the former *leaders*, and the latter *followers*. CSR first identifies who the leaders are; it then clusters them into communities with shared interests, based on their tagging activity. Users' queries (be them searches or recommendations) are then directed to the community of leaders who can best answer them. We have evaluated Clustered Social Ranking on the Web 2.0 CiteULike[1] website; our results demonstrate that CSR achieves an accuracy that is comparable to the best state-of-the-art recommender system techniques, but with a computational cost that is by orders of magnitude smaller, thus affording it better scalability in these fast growing settings.

The remainder of the paper is structured as follows: in Section 2 we review the state-of-the-art in recommender systems for the social web, highlighting their limitations in terms of cold-start problem. In Section 3 we present Clustered Social Ranking; Section 4 presents our experimental setup, in terms of dataset used, computed metrics, and benchmarks, while Section 5 analyses the obtained results. Finally, Section 6 concludes the paper.

[1] http://www.citeulike.org

2 Related Work

Research has been very active in the area of folksonomic searches and recommendations, splurged by the huge popularity of social tagging websites, such as Delicious, Flickr, Digg, Reddit, and the like. In these scenarios, users' tastes and interests are not expressed as numerical ratings, but rather as freely chosen tags associated to content. As a large study of social tagging conducted on the popular Delicious bookmarking system illustrated, folksonomies are so large and dynamic that traditional web search techniques are no longer affordable ([8]). Novel techniques, helping users to find relevant content in these settings, are thus called for.

One stream of research has focused on inferring the semantic relationship between tags, starting from an analysis of how users employ them. For example, [7] and [18] tried to build a navigable hierarchical taxonomy of tags, purely starting from tag usage. In [24], a simple technique to disambiguate tags is proposed, based on an analysis of the relationship between users, tags and resources. In [3], tag co-occurrence is broadly studied, starting from a tri-partite network of users, tags and resources; once again, the aim is to discover semantic relationships between tags, starting from information about how users associate them to resources.

A second stream of research has built upon the inferred relationship between tags to develop recommender system algorithms that assist users finding content of relevance within folksonomies. Some approaches have focused on mixed scenarios, where both numerical ratings and tags are available (e.g., [22,16]). Other approaches have been developed to target pure folksonomic settings, where users' preferences can only be inferred by analyzing their tagging activity. For example, [11] first identifies the best recommenders for a target user, based on what tags they have used in common (and how often); such user then receives as recommendations those items that have been most frequently tagged by them. FolkRank [9] uses a PageRank-like algorithm that employs the traditional random surfer model on the tri-partite graph of users-items-tags, producing very accurate recommendations in well connected networks.

One of the main issues left open by state-of-the-art tag-based recommender systems is the *cold-start problem*: when new users join the system, very little is known about their interests, so that predictions about what items they may like cannot be computed. This problem, well-known also in traditional recommender systems, appears to be aggravated in scenarios where likes and dislikes are not expressed as unambiguous numerical ratings, but rather as freely chosen tags. Some researchers have already moved in this direction: for example, [15] proposes to replace the old concept of users' similarity (which is not computable if users have not rated enough items in common), with a new concept of trust; in so doing, users make explicit who their trusted recommenders are. Such approaches are viable only in scenarios where bootstrapping a user's social network comes at no extra cost; moreover, the underpinning assumption that user's trust is a warranty of user's similarity limits the applicability of such approaches.

In the next section we present Clustered Social Ranking (CSR), our approach to tackle the cold-start problem both effectively and efficiently.

3 Clustered Social Ranking

In order to help new users of Web 2.0 websites find content of interest, we have built a technique that leverages upon the following two observations:

- **Leaders and Followers** - the vast majority of content on Web 2.0 websites is created by a rather small proportion of users (*leaders*), while the others mainly browse such content (*followers*). For example, according to an analysis of the CiteULike social bookmarking website, only 45% of the registered users actively posts items on the website, while the remaining 55% simply browse through other users' libraries [4]; this is confirmed by [25], whose analysis shows that more than 70% of the CiteULike users bookmark less than 10 resources overall.
- **Domains of Interest** - users tend to share interests with a rather small group of other users only. In a study of the CiteULike website [25], it was shown that even the most active users bookmark a rather tiny portion of the whole resource set; moreover, they use a rather small subset of the whole folksonomy, which they share with few other users only. This suggests that users have scoped interests that map to a small proportion of the whole social media content.

Clustered Social Ranking (CSR) exploits these observations as follow: rather than considering, as potential recommenders, the whole set of users within the Web 2.0 website, the much smaller set of leaders is considered (first observation). This is aligned with recent studies of more traditional recommender systems, where it was shown that accurate recommendations could be computed by narrowing the set of potential recommenders to the smaller set of users who have engaged the most with the system [1]. These leaders are clustered in domains of interest, based on their past tag activity (second observation). Whenever a user queries the system (be that a search initiated using an explicit set of tags, or a recommendation generated based on the tags used so far), CSR answers it by first identifying the community (or cluster) that can best answer it; it then relies on state-of-the-art tag-based recommender system techniques, applied within the cluster only, to rank content of interest to the user. We describe how leaders are identified and clustered in Section 3.1, while we illustrate how users' queries are dynamically associated to the best cluster to answer them in Section 3.2. Once a query has been associated to a cluster, we rely upon a previously developed algorithm (Social Ranking [25]) to finally compute an answer.

3.1 Clustering of Leaders

Based on the observation that the vast majority of content in Web 2.0 websites is actually produced by a very small proportion of users, Clustered Social Ranking

begins with the identification of so-called *leaders*, that is, users who have engaged with the system substantially more than average. Furthermore, based on the observation that users bookmark a rather tiny portion of the whole resource set, such leaders are *clustered* into domains of interest (i.e., users tagging the same resources, thus exhibiting similar interests). We decided to group users according to tagged content rather than considering the set of tags they used to avoid ambiguities that synonym or homonym tags may introduce. The rationale behind this *clustering of leaders* is that it should be much easier to find what cluster can best answer a query, among a small set (in the order of tens) of domain-focused clusters, rather than searching among tens of thousands of users who the best recommenders are, most especially so if the target user is new to the system. Leadership is simply defined in terms of activity: users who have tagged more resources than the average user within the system are elected.

The literature on clustering algorithms is very rich; we chose to experiment with the Fuzzy *c*-Means algorithm ([2]) because it has the very desirable property of each point having a degree of belonging to a cluster, rather than belonging completely to just one cluster. In our domain, this means that each user can be interested in multiple topics (i.e., belong to multiple clusters). Moreover, Fuzzy *c*-Means has a small computational complexity, which is linear in the number of existing clusters, in the number of items clustered and in the number of iterations performed (the latter being rather small, as we shall demonstrate later). To implement Fuzzy *c*-Means in our target scenario, we modeled each user u_i as a binary vector r_i over resources, where $r_i[j]$ is set equal to 1 if the user u_i has tagged resource p_j. k clusters are initially created, with k chosen following the empirical rule of thumb described in [12], that is, $k \approx (n/2)^{1/2}$, with n being the number of data points (in our case, users) to be clustered; each cluster is represented by a vector (called centroid) c_k, also modeled as a binary vector r_k over resources. The initialisation of such vectors was done by selecting, as centroids, the vectors of k real leaders with non-overlapping resource sets.

After this initialisation phase, Fuzzy *c*-Means performs a series of iterations in which each user is associated to one or more clusters, depending on how well the user is represented by the cluster she is being assigned to. In practice, this degree of belonging is computed as the cosine similarity between the user vector and each centroid vector:

$$sim(u_i, c_k) = cos(r_i, r_k) = \frac{r_i \cdot r_k}{||r_i|| * ||r_k||}$$

After each iteration, these values are normalized and fuzzyfied with a real parameter $m > 1$ so that their sum is 1 for each user; moreover, the centroid of each cluster is updated to be the mean of all users' vectors assigned to it, weighted by their degree of belonging to the cluster. This process is repeated until the algorithm has converged, that is, the change in the degree of belonging between two iterations is no more than a given sensitivity threshold.

Once the clustering of leaders has been completed, we maintain, for each cluster k, the following information:

- *Item Vector*: a vector R_k, where $R_k[j]$ counts how many users within the cluster have tagged resource j.
- *Tag Activity Vector*: a vector TA_k of all distinct tags used by users within k, whereby $TA_k[i]$ counts *how many times* tag t_i has been used to describe resources in R_k.
- *Tag Popularity Vector*: a vector TP_k of all distinct tags used by users within k, whereby $TP_k[i]$ counts *how many distinct users* within k have used tag t_i to bookmark items in R_k.

The above values have all been normalized in a $[0-1]$ range. In the next section, we explain how these vectors are being used to answer users' queries.

3.2 Answering Users' Queries

We use the term *user's query* q to represent both a (proactive) search and a (reactive) recommendation. The former represents the case whereby the user interacting with the system explicitly defines what she is looking for, by means of user-entered tags; the latter represents the case whereby the system recommends items to the user, based on all tags she has used so far. In the following, we do not distinguish between the two cases, and represent a user query q_u as a set of query tags $q_u = \{t_1, \ldots, t_n\}$. In order to answer such query, Clustered Social Ranking performs the following two steps:

(1) Query Association. First, CSR must find what cluster(s) can best answer q_u. To do so, it analysis the user's activity thus far with the system *and* the query tags. If the user has had little interaction with the system (i.e., she has tagged less than l resources, where l is not necessarily the same thresholding value used to define leaders), the query tags drive the association (*tag similarity association*). If the user has been actively engaged with the system instead, CSR further looks into the query tags: if $\{t_1, t_2, \ldots, t_n\}$ have been rarely used by u before (that is, they have been used less than the average tag usage for u), the association is driven once again by query tags; otherwise, it is driven by the user profile (*resource similarity association*). The underpinning idea is that, for users with a long history of interaction with the system, and querying the system within their well defined domain of interest, such history gives more information about what the best cluster is (i.e., who the best recommenders are) to answer a query; however, if the user is not well known to the system (cold-starter), or if indeed she is known, but she is currently looking for content outside her usual domain of interest, then the query tags are more informative of what she is after.

Association of users to cluster is then performed as follow: in the case of *tag similarity association*, we compute the cosine similarity between the query q_u and both the tag activity vector TA_k ($sim_{TA} = cos(q_u, TA_k)$) and the tag popularity vector TP_k ($sim_{TP} = cos(q_u, TP_k)$), for all clusters k. Groups are ranked based on $\max(sim_{TA}, sim_{TP})$, and those with a similarity higher than a given threshold elected to answer the query (in all our experiments, we used a threshold of zero as cosine similarity values in these high-dimensional spaces tend to be very low; we leave the exploration of alternative similarity measures

and thresholds for future work). Note that we use both TA and TP as these vectors provide complimentary information about the cluster: the former indicates how many *different resources* are a potential match for the query; the latter indicates how many *different users* within the cluster speak the same language of the query user (i.e., use the very same tags). In the case of *resource similarity association*, that is, the user is well known and her query tags correspond to her domain of interest, we compute the cosine similarity between her profile r_u (with $r_u[j]$ set equal to 1 if she has tagged resource j) and the item vector R_k (listing what resources have been tagged within cluster k), for all clusters k; as with tag association, groups are ranked based on $cos(r_u, R_k)$, and those with a similarity higher than a given threshold elected to answer the query (once again, in all our experiments, we used a threshold of zero). In both cases, if the similarity is zero towards all clusters, the query is associated to all of them; in practice, this means relying on all leaders to answer the query, regardless of their domain of interest. Note that, as leaders are substantially fewer than users, this is still much lighter than relying on the whole community, as traditional recommender system approaches do. Furthermore, in the experiments reported in the evaluation section, less than 3% of queries required associations to all groups.

(2) Resource Discovery and Ranking. Once the cluster(s) of suitable recommenders have been identified, Social Ranking (SR) [25] is used *within the cluster(s)* to discover and rank resources. In brief, Social Ranking works in two steps: when user u submits a query (be that a search or a recommendation) $q_u = \{t_1, t_2, \ldots, t_n\}$ to discover content that can be described by query tags $\{t_1, t_2, \ldots, t_n\}$, the set of query tags q_u is expanded so to include all $t_i \mid t_i \in q_u$ (for which $sim(t_i, t_i) = 1$), plus all tags $t_{n+1}, \ldots, t_{n+m}$ that are deemed similar to the query tags (for which $0 < sim(t_i, t_j) \leq 1$, with $i \in [1, n]$ and $j \in [n+1, n+m]$). Given tags t_i and t_j, tag similarity is computed as the cosine similarity of the tag vectors w_i and w_j, where $w_i[p]$ counts the number of times that tag t_i was associated to item p. After query expansion, all resources that have been tagged with at least one tag from the extended query set are retrieved. Their ranking depends on a combination of: the relevance of the tags associated to the resource with respect to the query tags (resources tagged with $t_i, i \in [1, n]$ should count more than those tagged with $t_j, j \in [n+1, n+m]$); and, the similarity of the taggers with respect to the querying user u (resources tagged by similar users should be ranked higher, as these users are more likely to share interests with u than others, and thus are in a better position to recommend relevant content). In CSR, rather than considering the similarity between the query user u and each user u_i within the selected cluster(s), we use the similarity computed during association. In so doing, the ranking of resources found *within* a cluster solely depends on the query tags; if the query is associated to more than one cluster, recommendations coming from the closest cluster are ranked higher than those coming from clusters with looser associations instead; to mark the difference further, we magnified the value of the query association by raising it to the power of a positive constant $\alpha > 1$. The rationale for this ranking process is the following: if the query user is a cold starter, or if she is

known to the system but with main interest in a different domain, computing user/user similarity would give meaningless values (in the former case) or misleading values (in the latter); in such case, only the query tags hold meaningful information for the ranking, and we further use the *tag similarity association* to better discriminate between resources coming from different clusters. If the user is well known to the system and she is seeking recommendations within her domain of interest, then user/user similarity should provide the same information given by the *resource similarity association*; we thus prefer the latter as it is cheaper to compute (one similarity computation per cluster instead of one per user within the cluster). The ranking of an item p found within cluster k would thus be computed as:

$$R(p) = \left(\sum_{t_j} sim(t_j, q^*_{u,k}) \right) * (sim_{ASSOC} + 1)^{\alpha}$$

where sim_{ASSOC} is the similarity computed during association between the query user u and the cluster k, and $q^*_{u,k}$ is the set of query tags expanded within k (i.e., using the tag similarity matrix of cluster k).

4 Simulation Setup

Having described the functioning of Clustered Social Ranking, we now describe how we have evaluated it. We begin with a presentation of the simulation setup: we define the metrics we have computed (Section 4.1), illustrate the dataset we have used (Section 4.2), and the benchmarks against which we have compared (Section 4.3). As CSR relies on a number of customisable parameters, we also discuss how these have been set (Section 4.4). We will then analyse the results obtained in Section 5.

4.1 Metrics

To evaluate the effectiveness of our query model, we adopted the standard Precision/Recall metrics computed at different cutting points of the recommendation list. More precisely:

$$Precision = \frac{|relevantContent| \cap |retrievedContent|}{|retrievedContent|}$$

$$Recall = \frac{|relevantContent| \cap |retrievedContent|}{|relevantContent|}$$

The former illustrates how much relevant content is retrieved, out of all content returned to the user; it thus gives a measure of how accurate the approach is. The latter computes how much relevant content is retrieved, out of all relevant content instead; it thus give a measure of coverage. Both metrics have been

computed after cutting the final recommendation list at the first 10, 20, 50, 100, 500, 1000 results retrieved.

To evaluate the scalability of our query model, we also analysed the computational complexity it entails, both in terms of online processing (operations computed whenever a query is issued) and offline processing (operations computed in batch mode, when the system is updated, for example, once a week/fortnight/month).

4.2 Dataset

We have conducted experiments using *CiteULike*, a social bookmarking website that aims to promote the sharing of scientific references. CiteULike enables scientists to organize their libraries with freely chosen tags which produce a folksonomy of academic interests. CiteULike runs a daily process which produces a snapshot summary of what articles have been posted by whom and with what tags up to that day. We downloaded one such archive in November 2009, containing bookmarks made between November 2004 to November 2009. We preprocessed the dataset to remove all non-alphabetical and non-numerical tags, following the same methodology proposed by the organisers of the ECML PKDD Discovery Challenge 2009[2]. The so-pruned archive contained 41,246 users, who had tagged 1,254,406 papers overall, using 210,385 distinct tags. To further remove excessive noise in the data, we used the p-Core preprocessing strategy, using the very same approach described in [5]. Based on this technique, both users, resources and tags are iteratively removed from the dataset, in order to produce a smaller but denser subset that guarantees each user, resource and tag to occur in at least p posts/bookmarks. We set $p = 5$; the final dataset contains 2,557 users, 7,480 papers, 3,153 tags, and 59,820 bookmarks.

During the experiments, we ordered the bookmarks according to the original date in which they were published, and we then performed a *temporal* split, so that the first 90% bookmarks have been used for training purposes, while the most recent 10% have been used for testing. We chose a temporal split, rather than a random one, to mimic the actual deployment of a social tagging website. On the training set, Clustered Social Ranking has been executed to pre-compute clusters and the associated information (see Section 3); each test bookmark has then been used as a query: the user who registered the bookmark is treated as the query user, and the tags associated to the bookmark as query tags. This information is given in input to CSR and a list of recommendations thus produced. The previously described metrics (i.e., precision and recall) have then been measured, considering as *relevant* the one resource to which the test bookmark refers to. Note that, because there is only one relevant resource we are after in the recommendation list (whose length has been cut from 10 to 1000), the measured precision is always very small; indeed, what is important is not the absolute precision value, but rather the precision that CSR entails relative to our benchmarks, described next.

[2] http://www.kde.cs.uni-kassel.de/ws/dc09/

4.3 Benchmarks

We have compared the precision/recall that Clustered Social Ranking achieves by comparing them with those of two benchmarks:

1. FolkRank *(FR)* - We have implemented the algorithm proposed by [9], which models the system as a weighted tri-partite graph where nodes refer to users, tags and resources. FR uses a random surfer strategy to recommend resources to users, following the idea that a resource that has been tagged with important tags and by important users becomes important itself. FolkRank is a state-of-the-art algorithm in tag-based recommender systems, whose accuracy has proved to be very high in dense Web 2.0 datasets.
2. Social Ranking *(SR)* - We have been comparing Clustered Social Ranking with the Social Ranking algorithm [25], where tag expansion is conducted by leveraging information from the whole community, and considering as potential recommenders any user of the system. Unlike FR, SR has shown high accuracy in sparse datasets; however, its computational overhead is non negligible in large folksonomies.

4.4 Parameters Tuning

Implementing Clustered Social Ranking requires the setting of a number of parameters. The first parameter refers to the thresholding value used to distinguish leaders from followers. In an actual deployment, this parameter would be set by dynamically studying the average bookmarking activity of users in the system, and by selecting a value above the average so to elect as leaders the smallest set of users who collectively tagged (almost) all resources in the system. In our datasets, we have used two thresholds: the first elects as leaders those users who have tagged more than 10 resources in the training set (shortly called UM10); the second selects as leaders those users tagging more than 30 items instead (shortly called UM30). Table 1 reports how many users are elected as leaders, and how many resources they have collectively bookmarked, with respect to the original dataset. Note that, when using a threshold of 30, less than 20% of users are elected as leaders, while only loosing less than 3% of bookmarked resources. This confirms the fact that a small portion of users is responsible for the vast majority of tagged content in the system; we thus expect that, by restricting our attention to this small set of users, coverage (i.e., recall) should not be hindered.

Table 1. Clusters Features

Dataset	Num. of Users	Num. of Resources	Num. of Tags
CiteULike	2484	7310	3137
CiteULike UM10	1189 (47%)	7291 (99%)	3056 (97%)
CiteULike UM30	432 (17%)	7116 (97%)	2811 (89%)

The second parameter affecting CSR is the number k of clusters. We have adopted the empirical rule of thumb described in [12], whereby $k \approx (n/2)^{1/2}$, with n being the number of data points (in our case, users) to be clustered. When considering leaders those who tagged more than 10 resources (CiteULike UM10), we used 26 clusters, to which we converged after 12 iterations; when clustering leaders who tagged more than 30 resources (CiteULike UM30), we used 14 clusters, to which we converged after just 5 iterations.

We set the remaining parameters required by CSR as follow: query expansion was limited to a maximum of $5 * m$ tags, with m being the number of query tags; upon query association, the minimum number of bookmarks l required for a user not to be considered in the cold start region was set to 10; finally, the α exponent used to mark differences between recommendations coming from clusters of different relevance was set to 5.

5 Results

We now present the results of our evaluation, focusing on effectiveness first (Section 5.1). As our approach is particularly geared towards new users, we present results subdivided in two groups: queries performed by active users, that is, those who have bookmarked at least 10 resources in the training set ($UM10$); and queries performed by new users, who have bookmarked less than 10 resources in the training set ($UL10$). In both cases, we discarded from the test set all queries for which the searched content does not belong to the training set, since none of the implemented algorithms would be able to answer such queries successfully. Of the remaining 4,575 test bookmarks (i.e., our queries), 3,156 were done by active users ($UM10$) and 1,419 by non active ones ($UL10$).

5.1 Evaluation of Effectiveness: Precision and Recall

As shown in Figure 1, Clustered Social Ranking (CSRUM10 and CSRUM30), Social Ranking and FolkRank all achieve very similar precision and recall for active users (with (C)SR being slightly better than FR).

We now turn our attention to new users instead. As Figure 2 illustrates, these users are much more difficult to predict, and even an advanced query engine like FolkRank is not capable of computing valuable recommendations, as too little information is available about these users. However, CSR is capable of exploiting the little information available in the query and about leaders to produce a precision and recall which are comparable to those of SR, and 28% and 40% respectively better than those provided by FR (for recommendation lists of 50 elements). We can thus conclude that, when considering active users, both SR, CSR and FR have very similar performance, both in terms of precision and recall. When considering cold-start users instead, SR and CSR are the most effective techniques, with a neat gain over FR. As we shall discuss next, the computational cost that CSR entails is sensibly lower than both FR and SR, thus making it the most suitable approach overall in scenarios where both effectiveness and efficiency equally matter.

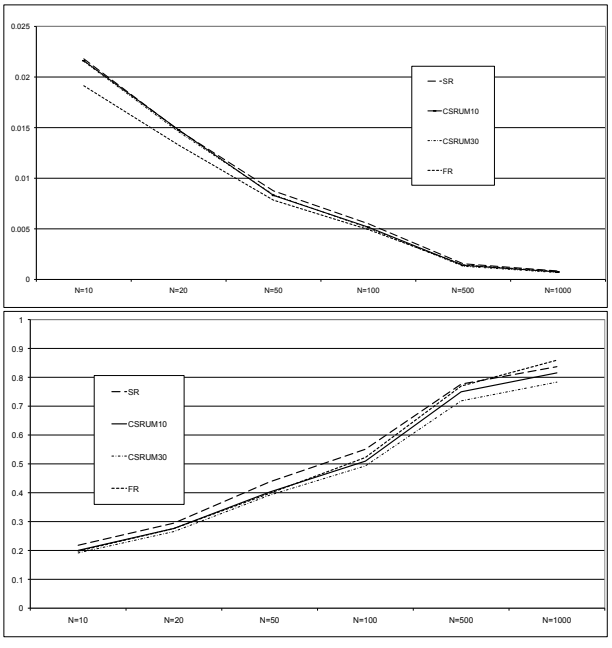


Fig. 1. Precision/Recall for Active Users on CiteULike

5.2 Evaluation of Efficiency: Computational Complexity Analysis

In this section, we analyse the computational complexity that FR, SR and CSR entail, in order to demonstrate that CSR is the most lightweight approach, with a computational cost which is by orders of magnitude lower than that entailed by the other techniques.

In quantifying the computational cost of such approaches, we distinguish between *offline* cost, that is, the cost entailed to pre-compute all the data structures the algorithms rely on (for example, the matrix of users' similarities). Typically, recommender systems recompute these values periodically (e.g., weekly, fortnightly, monthly); there is a tension between accuracy and scalability: the more often the update process is run, the more accurate the recommendations computed, but also the higher the computational cost entailed. For each approach, we will also quantify the *online* computational cost of executing each query. Table 2 reports the computational complexity of each approach.

FolkRank requires no offline pre-computation; rather, it maintains the tripartite graph of users, resources and tags live. Whenever a new query arrives, FR traverses such graph using i iterations (typically 30-35), computes a score for all resources, and derives a recommendation list based on such scores. If we indicate with Y the number of arches in this graph, the online cost entailed by FolkRank is thus $O(i * Y)$.

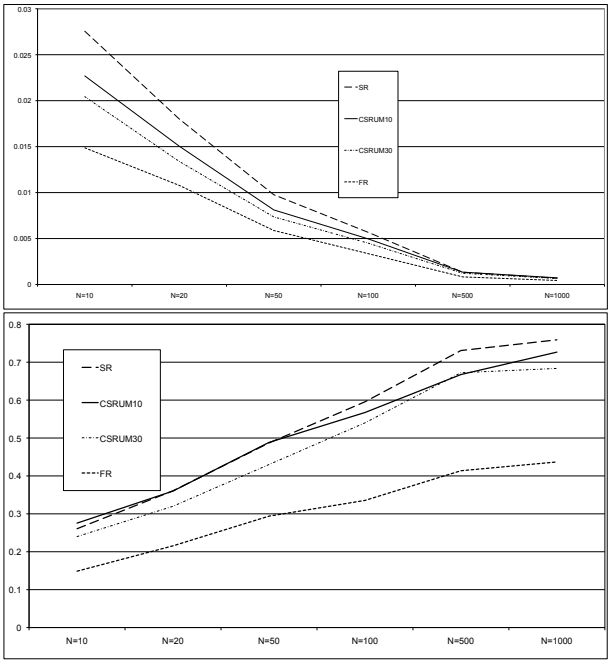


Fig. 2. Precision/Recall for New Users on CiteULike

Table 2. Computational Complexity of FR, SR and CSR

Approach	**Offline**	**Online (per query)**	**OfflineCUL**	**OnlineCUL (all queries)**
FR	-	$O(i * Y)$	-	23,000 Million
SR	$O\left(\frac{U*(U-1)}{2} + \frac{T*(T-1)}{2}\right)$	$O(R * T)$	8 Million	10 Million
CSR	$O\left(i * k * u + k * \frac{t*(t-1)}{2}\right)$	$O(k * r * t)$	1.5 Million	4 Million

Social Ranking requires the offline computation of two matrices: one storing users' similarity, and another storing tags' similarity. These matrices are symmetric, thus its offline cost is $O(U * (U-1)/2 + T * (T-1)/2)$, with U being the number of users in the system and T the number of tags. The online cost depends instead on the number of resources and tags which have to be taken into account in order to answer each user query; in the worst case, all resources R have been tagged with all tags T in the system, thus $O(R * T)$.

Clustered Social Ranking requires two offline computations: the execution of the Fuzzy c-Means Algorithm to cluster leaders, and the computation of the tags' similarity matrices for each cluster. The former is linear in the number of users to cluster u, the number of iterations i required to converge, and the number of clusters k [10]. The latter is a symmetric matrix, so if we indicate with t the number of tags used within each cluster, the offline cost of CSR is

$O(i * k * u + k * t * (t-1)/2)$. The online computational complexity can be estimated as $O(k * r * t)$, in the worst case where the query is associated to all clusters k, within which all resources r have been tagged, with all tags t.

These theoretical limits are upper bounds on the actual complexity of each approach. For example, queries in SR/CSR never require all R/r resources and T/t tags to be answered, nor are they associated to all k clusters. To give a flavour of the actual cost entailed by each approach in a real deployment, we have computed their offline and online cost, when answering all $4,575$ queries from the CiteULike UM30 dataset. The results are reported in the last two columns of Table 2. As shown, the online cost of FolkRank is prohibitive in real deployments, despite the fact that the actual size of the tri-partite graph Y is much smaller (i.e., sparser) than a full one. FR's main disadvantage is that it requires a complete computation of the Page Rank vector for each query, making it unsuitable to work with data from large Folksonomies (as also confirmed by [5]).

Both CSR and SR amortize the cost of pre-computing tags' and users' similarities, affording a much smaller computational cost overall (offline + online); note that, for example, the offline processing of SR could be repeated once every other query in the test set, and still be computationally cheaper than FR (in practice, several thousands queries are normally answered within a single offline update). We now take a closer look at CSR versus SR. The online cost of CSR is half that of SR. More importantly, the offline cost is an order of magnitude smaller; this is because, in practice, $u << U$ (leaders are much fewer than all users), so the cost of clustering them is much smaller than computing users' similarity. For example, in CiteULike UM30, there are only 432 leaders, as opposed to 2,484 users overall: the cost of clustering leaders is only 30K computations (with $k = 14$ clusters and $i = 5$ iterations to converge), while the cost of quantifying users' similarity is 3M. Moreover, each cluster has a much smaller number of tags, so that, even if a separate tags' similarity matrix has to be computed for each cluster, their overall cost (1.5M computations) is much smaller than that entailed by the single matrix maintained by SR (5M computations). The neat reduction in the offline cost of CSR also means that such data structures can be re-computed more often that those used by SR, thus being able to achieve higher accuracy without compromising scalability. Note that frequent updates are of paramount importance in rapidly growing settings and especially for new users, where one update more can make the difference between knowing a little about the user's preferences (her first few bookmarks) or nothing at all.

6 Conclusion

In this paper, we have presented Clustered Social Ranking, a novel search and recommendation technique specifically developed to support new users of Web 2.0 websites finding content of interest. CSR exploits the fact that the vast majority of content on Web 2.0 websites is created by a small proportion of users, while the others mainly browse such content. CSR first identifies who the leaders

are, clusters them into communities with shared interests, and subsequently answers users' queries (be them searches or recommendations) by directing them towards the community of leaders who can best answer them. Our evaluation conducted on the CiteULike website demonstrates that CSR achieves high accuracy, while entailing a low computational cost, thus making it the most suitable solution in these fast growing settings.

Acknowledgement. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under the Grant Agreement n. 234239. The authors are solely responsible for it and it does not represent the opinion of the Community. The Community is not responsible for any use that might be made of information contained therein.

References

1. Amatrian, X., Lathia, N., Pujol, J.M., Kwak, H., Oliver, N.: The wisdom of the few. In: Proc. of SIGIR, Boston, Massachusetts (2009)
2. Bezdek, J.C.: Pattern recognition with fuzzy objective function. Kluwer Academic Publishers, USA Press, Norwell, MA (1981)
3. Capocci, A., Caldarelli, G.: Folksonomies and clustering in the collaborative system citeulike. Journal of Physics A: Mathematical and Theoretical 41(22), 224016–224023 (2008)
4. Emamy, K., Cameron, R.: Citeulike: A researcher's social bookmarking service (2007)
5. Gemmell, J., Schimoler, T., Ramezani, M., Mobasher, B.: Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies. In: Proc. of 7th Intelligent Techniques for Web Personalization and Recommender Systems Workshop, vol. 528 (2009)
6. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. In: Proc. of SIGIR, pp. 230–237. ACM Press, Berkley (1999)
7. Heymann, P., Garcia-Molina, H.: Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems. Technical Report 2006-10, Stanford University (April 2006)
8. Heymann, P., Koutrika, G., Garcia-Molina, H.: Can social bookmarking improve web search? In: Proc. of the International Conference on Web Search and Web Data Mining, pp. 195–206. ACM, New York (2008)
9. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folksonomies: Search and Ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS (LNAI), vol. 4011, pp. 411–426. Springer, Heidelberg (2006)
10. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31(3), 264–323 (1999)
11. Ji, A.-T., Yeon, C., Kim, H.-N., Jo, G.-S.: Collaborative tagging in recommender systems. In: Proc. of Advances in Artificial Intelligence, pp. 377–386. ACM, New York (2007)
12. Mardia, K.V., Kent, J.T., Bibby, J.M.: Multivariate analysis. Academic Press, London (1979)

13. Lam, X.N., Vu, T., Le, T.D., Duong, A.D.: Addressing cold-start problem in recommendation systems. In: Proc. of the 2nd International Conference on Ubiquitous Information Management and Communication, pp. 208–211. ACM, New York (2008)
14. Leung, C.W.-k., Chan, S.C.-f., Chung, F.-l.: Applying cross-level association rule mining to cold-start recommendations. In: Proc. of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops, pp. 133–136. IEEE Computer Society, Washington, DC, USA (2007)
15. Massa, P., Avesani, P.: Trust-aware recommender systems. In: Proc. of Conference on Recommender Systems, pp. 17–24. ACM, New York (2007)
16. Nakamoto, R., Nakajima, S., Miyazaki, J., Uemura, S.: Tag-based Contextual Collaborative Filtering. In: 18th IEICE Data Engineering Workshop (2007)
17. Papagelis, M., Rousidis, I., Plexousakis, D., Theoharopoulos, E.: Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 553–561. Springer, Heidelberg (2005)
18. Passant, A.: Using Ontologies to Strengthen Folksonomies and Enrich Information Retrieval in Weblogs. In: Proc. of International Conference on Weblogs and Social Media (2007)
19. Peng, H., Bo, X., Fan, Y., Ruimin, S.: A scalable p2p recommender system based on distributed collaborative filtering. Expert Systems with Applications (2004)
20. Schein, A., Popescul, A., Ungar, L., Pennock, D.: Methods and metrics for cold-start recommendations. In: Proc. of the 25th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 253–260 (2002)
21. Sen, S., Lam, S.K., Rashid, A.M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, M.F., Riedl, J.: Tagging, communities, vocabulary, evolution. In: Proc. of the 20th Conference on Computer Supported Cooperative Work, pp. 181–190. ACM Press, New York (2006)
22. Tso-Sutter, K.H.L., Marinho, L.B., Schmidt-Thieme, L.: Tag-aware Recommender Systems by Fusion of Collaborative Filtering Algorithms. In: Proc. of 23rd Annual Symposium on Applied Computing, pp. 16–20. ACM Press, New York (2008)
23. Wu, X., Zhang, L., Yu, Y.: Exploring social annotations for the semantic web. In: Proc. of the 15th International Conference on World Wide Web, pp. 417–426. ACM Press, New York (2006)
24. Yeung, C.-m.A., Gibbins, N., Shadbolt, N.: Mutual Contextualization in Tripartite Graphs of Folksonomies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 966–970. Springer, Heidelberg (2007)
25. Zanardi, V., Capra, L.: Social ranking: uncovering relevant content using tag-based recommender systems. In: Proc. of the ACM Conference on Recommender systems, New York, NY, USA, pp. 51–58 (2008)

Author Index